UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Teoría de la Señal y Comunicaciones

Ing. Téc. de Telecomunicaiones: Sonido e Imagen

PROYECTO FIN DE CARRERA

Rediseño y Ampliación de 'IMAGine': Web con Cursos Interactivos de Tratamiento Digital de Imagen

AUTORA: *Ana Belén Alonso Martín (NIA: 100048581)*

TUTOR: *Jesús Cid Sueiro*

Octubre 2008

PROYECTO FIN DE CARRERA

Ing. Tec. Telec. Especialidad Sonido e Imagen

REDISEÑO Y AMPLIACIÓN DE IMAGine: WEB con Cursos Interactivos de Tratamiento Digital de Imagen.

Autora: **Ana Belén Alonso Martín** (100048581).

Tutor: Jesús Cid Sueiro.

Departamento: Teoría de la Señal y Comunicaciones.

TRIBUNAL CALIFICADOR:

Presidente:

Secretario:

Vocal:

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día **16 de Octubre de 2008** en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la

CALIFICACIÓN de: **MH (Matrícula de Honor)**

Presidente	Secretario	Vocal
Ángel Navia Vázquez	Sergio Sanz Rodriguez-Escalona	Ángel García Olaya

Leganés, **16 de Octubre de 2008**

Dedicatorias:

A mis padres y hermanos,
por estar siempre a mi lado

A mis primas,
por escucharme y hacerme reir
(Sara, Cristina… este proyecto
no habría sido posible sin vosotras)

A los amigos,
por tenderme una mano
siempre que lo he necesitado

A Lola,
por ser la mejor compañera de prácticas
en la carrera y sobre todo por ser una gran amiga

Especialmente a Javi,
por su apoyo incondicional y su infinita paciencia

Y por último, a Jesús Cid,
por ofrecerme la oportunidad de hacer mi
proyecto sobre algo que realmente me apasiona

RESUMEN DEL PROYECTO:

IMAGine es una web [disponible en la URL: http://www.tsc.uc3m.es/imagine/] con cursos interactivos de tratamiento digital de imagen, creada a partir de varios proyecto fin de carrera anteriores. Incluye elementos interactivos (applets) desarrollados en lenguaje de programación Java.

En este proyecto se ha realizado una reestructuración completa de IMAGIine, creando la versión 2.0. de la aplicación. Dicha reestructuración ha abarcado tanto a la parte estética (la propia apariencia de la web, compatibilidad entre distintos navegadores…), como a la parte técnica (inclusión de la teoría de los cursos en formato web, implementación de versiones web/docente/impresión para optimizar la utilidad de la aplicación con fines docentes, creación de autoevaluaciones para facilitar el aprendizaje del alumno…). Además, se ha ampliado IMAGine con un nuevo curso, el de 'Restauración de Imágenes'.

En la parte relativa a los applets, se ha modificado, estructurado y homogeneizado el código java que los implementan. Se han creado los applets del curso de restauración de imágenes y se ha añadido la funcionalidad de tratar imágenes a color en ellos, incluyendo un menú con las opciones básicas de procesado de imágenes a color.

ABSTRACT:

IMAGine is a web site (available at the url http://www.tsc.uc3m.es/imagine/) which includes interactive digital image editing courses. It is the result of the combined effort of several previous bachelor thesis. It also includes interactive elements (applets) developped in Java programming language.

This bachelor thesis has consisted of a complete refactoring of this website, which has meant to create its 2.0 version, including the aesthetic part of the website (look, compatibility among the most popular internet browsers, etc) as well as the technical part (including the theoretical part of the courses in web files, implemeenting web/teaching/printing version of each of them to optimize the usefulness of the application int teaching terms, designing auto-evaluations to make easy the pupil learning, etc).

In addition, IMAGine has been completed with a brand new course, called 'Image restoring', which included several new applets.

Finally, all the functionality related with the applets, it has been completely modified to restructure and homogenize the java implementation, and a new functionality for editing colour images has been added to all of them.

PALABRAS CLAVE:

(Cursos interactivos) Tratamiento Digital Imágenes, Transformadas Imágenes, Procesado Morfológico, Restauración Imágenes, Applets tratamiento imágenes…

ÍNDICE

1. INTRODUCCIÓN 13

1.1 ¿QUÉ ERA IMAGINE ANTES DE ESTE PROYECTO? 15

1.2 ¿QUÉ ES IMAGINE AHORA? 17

1.3 CONTENIDO DE LA MEMORIA 18

2. ESTADO DEL ARTE 21

2.1. INTRODUCCIÓN 21

2.1.1. Introducción al e-learning (aprendizaje basado en web) 21

2.2. CURSOS DE TRATAMIENTO DIGITAL DE IMAGEN EN INTERNET 24

2.2.1. Curso interactivo de Trat. Digital de Imagen (Universidad de Málaga) 25
2.2.2. HIPR2: Image Processing Learning Resources (Universidad de Edimburgo) 28
2.2.3. Técnicas de Procesado de Imagen (Universidad de la Coruña) 33
2.2.4. Otros cursos 36

2.3. CONCLUSIONES 39

3. TEORÍA SOBRE TRATAMIENTO DIGITAL DE IMÁGENES 43

3.1. INTRODUCCIÓN AL TRATAMIENTO DIGITAL DE IMÁGENES 43

3.2. RESTAURACIÓN DE IMÁGENES 49

3.2.1. Introducción a la Restauración de Imágenes 49
3.2.2. Técnicas de mejora: casos de restauración 50
3.2.3. Técnicas de restauración: restauración basada en modelos 55
3.2.4. Aplicaciones de la Restauración de Imágenes 59

4. DESCRIPCIÓN DE LA HERRAMIENTA 63

4.1. APLICACIÓN WEB 63

4.1.1. WEB: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES 63

4.1.2. WEB: FUNCIONAMIENTO A NIVEL DE USUARIO 65

4.1.2.1. Parte 1: Portada de IMAGine y Apartados no relacionados con cursos y autoevaluaciones 67
4.1.2.2. Parte 2: Cursos y autoevaluaciones 69

4.1.3. WEB: FUNCIONAMIENTO A NIVEL TÉCNICO 71

4.1.3.1. Introducción a las tecnologías implicadas en la parte web 75
4.1.3.2. Funcionamiento técnico de la portada de IMAGine y resto de enlaces del índice no relacionados con cursos y autoevaluaciones 79
4.1.3.3. Funcionamiento técnico de los cursos 81
4.1.3.4. Funcionamiento técnico de las autoevaluaciones 88

4.1.4. WEB: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA 90

4.1.4.1. Crear una nueva página en la parte de la portada....91
4.1.4.2. Crear un curso nuevo....91
4.1.4.3. Crear una autoevaluación nueva dentro de un curso existente....93
4.1.4.4. Crear una autoevaluación nueva de un curso nuevo....94

4.2. APPLETS....95

4.2.1. APPLETS: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES....95

4.2.2. APPLETS: FUNCIONAMIENTO A NIVEL DE USUARIO....100

4.2.2.1. APPLETS DEL CURSO 'PROCESADO BÁSICO DE IMG'....100
4.2.2.1.1. Applet Básico de Imágenes a escala de grises....100
4.2.2.1.2. Applet "Operaciones Puntuales"....101
4.2.2.1.3. Applet "Filtrado Basado en Máscaras"....102
4.2.2.1.4. Applet "Importancia Componentes de Baja Frecuencia"....102
4.2.2.1.5. Applet "Filtro de Mediana"....103
4.2.2.1.6. Applet "Rotación de Imágenes y su efecto en la DFT"....104
4.2.2.1.7. Applet "Contribución de las componentes frecuenciales en la DFT"....106
4.2.2.1.8. Applet "Importancia de la fase"....106
4.2.2.1.9. Applet "Experimento del Oppenheim"....107
4.2.2.1.10. Applet "Promedio de módulos y fases"....108

4.2.2.2. APPLETS DEL CURSO 'PROCESADO MORFOLÓGICO'....109
4.2.2.2.1. Applet "Operaciones morfológicas sobre imágenes binarias"....109
4.2.2.2.2. Applet "Transformación Hit or Miss (Acierta/Falla)"....110
4.2.2.2.3. Applet "Algoritmos morfológicos"....112
4.2.2.2.4. Applet "Operaciones morfológicas sobre imágenes en escala de grises"....113

4.2.2.3. APPLETS CURSO 'RESTAURACIÓN DE IMÁGENES'....113
4.2.2.3.1. Applet "Ecualización del Histograma"....114
4.2.2.3.2. Applet "Cancelación de Interferencias Sinusoidales"....114
4.2.2.3.3. Applet "Eliminación de Ruido"....115
4.2.2.3.4. Applet "Distorsiones Geométricas"....116
4.2.2.3.5. Applet "Imágenes en color"....117

4.2.3. APPLETS: FUNCIONAMIENTO A NIVEL TÉCNICO....119

4.2.3.1. Package BaseImagenes: Clases que se ocupan de la base de imágenes....119
4.2.3.2. Package comun: Clases que contienen la funcionalidad básica de la aplicación....124
4.2.3.3. Package de los applets de los cursos....135

4.2.4. APPLETS: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA....140

4.2.4.1. Añadir un nuevo curso a IMAGine....140
4.2.4.2. Añadir un nuevo applet con imágenes en grises a un curso....140
4.2.4.3. Añadir un nuevo applet con imágenes a color a un curso....141
4.2.4.4. Añadir una técnica de procesado nueva al menú emergente de imágenes a escala de grises 141
4.2.4.5. Añadir una técnica de procesado nueva al menú emergente de imágenes a color....142

5. ESTADÍSTICAS DE TRÁFICO RECIBIDO 145

6. CONCLUSIONES 151

7. LINEAS FUTURAS 155

8. ANEXOS 161

8.1. CLASES MÁS IMPORTANTES DE LA AMPLIACIÓN A IMÁGENES A COLOR....161

8.1.1. Clase ImagenColor *161*
8.1.2. Clase ImagenColorCanvasCtes *170*
8.1.3. Clase ImagenColorCanvas *171*
8.1.4. Clase FFTImagenColor *178*
8.1.5. Clase CargaImagenesCtes *183*
8.1.6. Clase CargaImagenes *184*
8.1.7. Clase Editor_BaseImgColor *189*

8.2. CLASES MÁS IMPORTANTES DEL CURSO RESTAURACIÓN DE IMÁGENES 196

8.2.1. Clase Imagen *196*
8.2.2. Clase RestauracionImg *209*

9. ÍNDICE DE FIGURAS _____ 219

10. BIBLIOGRAFÍA _____ 225

1. INTRODUCCIÓN

1. INTRODUCCIÓN ____ **15**

1.1 ¿QUÉ ERA IMAGINE ANTES DE ESTE PROYECTO? ____ 15

1.2 ¿QUÉ ES IMAGINE AHORA? ____ 17

1.3 CONTENIDO DE LA MEMORIA ____ 18

1. INTRODUCCIÓN

Este proyecto fin de carrera, constituye un "Rediseño y Ampliación de IMAGine: Web con Cursos Interactivos sobre Tratamiento Digital de Imagen".

1.1 ¿QUÉ ERA IMAGine ANTES DE ESTE PROYECTO?

Empezaremos describiendo que era IMAGine antes de este proyecto. En un principio, se pensó en realizar una web donde se ofreciese un pequeño curso on-line sobre los fundamentos del tratamiento digital de imagen, para que profesores y alumnos de asignaturas relacionadas con esta materia contasen con una eficaz herramienta de apoyo que les permitiese ilustrar algunos ejemplos prácticos (con elementos interactivos, los applets), y que también fuese accesible para el resto del público en general.

Tras esta idea inicial, se han ido sucediendo distintos proyectos fin de carrera que han ido ampliando y mejorando el curso inicial, haciendo de IMAGine una de las páginas en español mejor posicionadas en el ranking de los principales buscadores de Internet en cuanto a Cursos Interactivos de Tratamiento Digital de Imagen.

A continuación, ofrecemos un breve resumen de la historia de IMAGine, con los autores que han contribuido en ella y su aportación realizada:

1998. Versión 1.0.
Applet Imagen y Curso Básico.
Autor: Pablo Puente Domínguez. Universidad de Valladolid.

2005. Versión 1.1.
Depuración de código y mejoras sobre el applet de carga de imágenes.
Autor: Fernando Hoyos Leyva. Universidad Carlos III de Madrid.

2006. Versión 1.2.
Curso de Procesado Morfológico.
Autora: Marta Monsálvez Elías. Universidad Carlos III de Madrid.

2008. Versión 2.0.
Curso de Restauración de Imágenes.
Reestructuración del diseño de todo IMAGine en su parte web y java:
-Parte web: Nuevo diseño de la página, compatibilidad entre distintos navegadores, integración de la parte teórica en formato web, vistas web/docente/impresion, autoevaluaciones, ...
-Parte java (applets): Unificación y reestructuración del código existente. Extensión a imágenes a color e implementación de sus funcionalidades principales, con aplicación de carga de imágenes. Applets del curso de Restauración de imágenes.
Autora: Ana Belén Alonso Martín. Universidad Carlos III de Madrid.

Coordinación: Jesús Cid Sueiro
Profesor Titular de la Universidad Carlos III de Madrid, Departamento de Teoría de la Señal.

Figura 1.1: Historia de IMAGine: versiones, autores y contribuciones.

No obstante, la última versión de IMAGine antes de este proyecto (versión 1.2) seguía presentando ciertas limitaciones que sugerían la necesidad de una nueva versión que sentase las bases definitivas del proyecto, y no que fuese una mera continuación o ampliación de proyectos anteriores.

• *En cuanto a la parte web, las limitaciones principales consistían en:*

- Falta de una interfaz adecuada y atractiva para el usuario, que permitiese una navegación fácil, rápida e intuitiva por todo el contenido del curso.

- Adaptación de la web para su correcta visualización en cualquiera de los principales navegadores.

- Mala integración del contenido teórico de la asignatura dentro de la web: En la versión 1.2 simplemente incluía las presentaciones en PowerPoint del profesor en formato flash (sufriendo la consiguiente pérdida de calidad de las imágenes, empeorando con ello su legibilidad), y ni siquiera se podía avanzar/retroceder dinámicamente entre las distintas diapositivas. Por tanto, escasa orientación docente de los cursos (un profesor no podía impartir la teoría de una clase utilizando la web como apoyo, debido a las dificultades que mencionábamos, sino que simplemente se podía apoyar en la web para acceder a los applets utilizándolos como ejemplos ilustrativos de ciertos apartados).

- Inexistencia de herramientas de autoevaluación que permitiesen al usuario medir los conocimientos adquiridos en los cursos de IMAGine.

- Falta de una herramienta eficaz para medir estadísticas de uso de la web (existía un contador para medir el número de visitas, pero no una herramienta de medición de estadísticas que permitiese obtener datos tales como la procedencia de las visitas, el medio de acceso la web (buscador, enlace desde otra web), que navegador y resolución utilizaban, histórico del nº de visitas por días y horas…).

- Escasez de cursos que abarcaban solo un pequeño abanico de temas relacionados con tratamiento digital de imagen.

• *En cuanto a la parte java (código de los applets), los principales inconvenientes eran:*

- Desunificación de código procedente de los distintos proyectos anteriores, siguiendo distintas reglas y estructuras en cada uno de ellos, dificultando con ello ampliaciones futuras (compleja e intrincada red de clases difíciles de comprender).

- Nombres de clases, métodos y variables poco descriptivos de su utilidad, que junto al excesivo número de clases (innecesarias en muchos casos, ya que por ejemplo se creaban una clase nueva para hacer un simple Panel) hacían de IMAGine una herramienta de difícil comprensión.

- Pequeños fallos en algunos de los applets (en los applets de Procesado Morfológico, al cargar una nueva imagen, no se ponían en blanco el resto de contenedores del applet encargados de mostrar los resultados del procesado).

1.2 ¿QUÉ ES IMAGine AHORA?

Con este proyecto, se ha pretendido unificar el contenido de los proyectos fin de carrera anteriores reestructurando y rediseñando IMAGine (que ya empieza a ser una aplicación bastante grande), para facilitar la tarea de futuras ampliaciones de la herramienta.

Esta *reestructuración* ha sido tanto a nivel de web (se ha creado una web totalmente nueva que incluye muchas más posibilidades), como a nivel del código java de los applets (se han mantenido los applets existentes de cursos anteriores, pero se ha rediseñado totalmente toda la estructura de clases java que los implementaban primando la sencillez de código y la estructuración similar de todos los appets, entre otras cosas).

Con la creación de la nueva web, hemos perseguido los siguientes objetivos:

- Construir una aplicación web con Cursos INTERACTIVOS sobre Tratamiento Digital de Imagen construidos mediante hipertexto (XHTML 1.0 que facilita la accesibilidad web), imágenes apoyando la teoría, animaciones ilustrando algunos efectos y herramientas interactivas implementadas con applets que permitan la experimentación de ciertas técnicas de procesado de imágenes. Todo ello orientado a facilitar el aprendizaje de la materia al mayor público posible.

- Disponer de una herramienta propia, diseñada en base a la asignatura de Tratamiento Digital de Imagen impartida en la Universidad Carlos III (creando un curso mucho más específico a las necesidades de esta asignatura y por tanto más útil para impartirla utilizando esta herramienta), que elimina la dependencia de fuentes externas (cursos más o menos similares existentes en la web) que no garantizan la disponibilidad de los recursos.

- Hacer de esta aplicación una herramienta mucho más orientada al uso docente, pero sin descuidar tampoco la accesibilidad de cualquier usuario de Internet a la web. Para ello se ha insertado en formato web la teoría de la asignatura de Tratamiento Digital de Imagen impartida en la Universidad Carlos III por el tutor de este proyecto en cada uno de los cursos; se han diseñado tres versiones de la web según el perfil del usuario que la visite (versión web, docente y de impresión); se han incluido ejercicios y un apartado con autoevaluaciones con formularios de autoevaluación de cada uno de los cursos existentes en la actualidad.

Por otro lado, la *ampliación* de IMAGine ha consistido en la creación de un curso nuevo, el de Restauración de Imágenes, tanto a nivel de teoría como creando los applets que servirán de apoyo al curso.

Se ha elegido el curso de Restauración de Imágenes porque es un tema sobre el que todavía queda mucho por investigar, dado que aún no se ha encontrado la metodología idónea para resolver este problema de manera genérica, sino que simplemente se han encontrado soluciones específicas para restaurar imágenes con una determinada tipología. Por tanto, este tema ofrece amplias posibilidades tanto a nivel teórico como a nivel práctico, permitiéndonos diseñar applets con los que poder experimentar distintas técnicas de restauración, observando ventajas y desventajas de cada una de ellas.

En cuanto a la ampliación de los applets, además de las creación de los applets del curso de Restauración, se ha añadido la funcionalidad para que se puedan crear applets con imágenes a color (hasta ahora solo se podían hacer applets con imágenes a escala de grises), incluyendo un submenú propio para este tipo de imágenes que nos permita hacer ciertas operaciones (mostrar histograma de las componentes R, G, B; filtrado; rotación…).

Con todas estas actuaciones, IMAGine se constituye como uno de los cursos interactivos sobre tratamiento digital de imagen más completos de los existentes en la web, siendo una herramienta real de apoyo a la docencia (incluso se podría proyectar la web en una clase e impartir dicha clase apoyándose en ella), pero también siendo una herramienta de gran utilidad para el estudio de la materia por cualquier persona, por su carácter autocontenido.

No obstante, hay ciertos temas de procesado de imagen que aún no están incluidos en la web (Segmentación, Visión estereoscópica, etc.) y se escapan a lo temas abarcados en este Proyecto, pero que sin duda en futuras ampliaciones de la herramienta serán implementados completando totalmente el abanico de temas de Tratamiento Digital de Imagen.

1.3 CONTENIDO DE LA MEMORIA

Por último, y centrándonos en el contenido de esta memoria, a continuación describimos el contenido de los capítulos que la componen:

- Capítulo 2: Investigación sobre el panorama existente en Internet sobre Cursos de Tratamiento Digital de Imagen.

- Capítulo 3: Introducción teórica referente al tema del procesado de imágenes, centrándonos especialmente en el tema de la restauración de imágenes.

- Capítulo 4: Análisis detallado de todo IMAGine, tanto de la parte Web como de la parte Java que constituyen los applets, ofreciendo la perspectiva de su funcionamiento a nivel de usuario pero dando también una visión técnica de la aplicación. Para facilitar la ampliación de la herramienta en futuros proyectos fin de carrera, además incluimos una guía rápida con las pautas básicas necesarias para ello.

- Capítulo 5: Estadísticas obtenidas con el Google Analytics sobre el tráfico recibido en la url de IMAGine (http://www.tsc.uc3m.es/imagine).

- Capítulo 6 y 7: Se establecen las conclusiones y se apuntan las siguientes líneas de ampliación de IMAGine.

- El resto de capítulos (8, 9 y 10) ofrecen información adicional a este proyecto, siendo el 8 un anexo con el código de las clases Java más importantes de IMAGine, el 9 un índice de las figuras incluidas en esta memoria y el 10 un índice con referencias bibliográficas consultadas en la elaboración de este proyecto.

2. ESTADO DEL ARTE

2. ESTADO DEL ARTE 21

2.1. INTRODUCCIÓN 21

2.1.1. Introducción al e-learning (aprendizaje basado en web) *21*

2.2. CURSOS DE TRATAMIENTO DIGITAL DE IMAGEN EN INTERNET 24

2.2.1. Curso interactivo de Trat. Digital de Imagen (Universidad de Málaga) *25*
2.2.2. HIPR2: Image Processing Learning Resources (Universidad de Edimburgo) *28*
2.2.3. Técnicas de Procesado de Imagen (Universidad de la Coruña) *33*
2.2.4. Otros cursos *36*

2.3. CONCLUSIONES 39

2. ESTADO DEL ARTE

2.1. INTRODUCCIÓN

Antes de diseñar una herramienta basada en web, es conveniente analizar los recursos existentes en Internet. Con ello podremos saber si realmente las necesidades de la aplicación que pretendemos crear están ya cubiertas o si por el contrario nada de lo existente abarca nuestros objetivos.

Nuestra aplicación, IMAGine, pretende ser un curso on-line sobre tratamiento digital de imagen orientado a cualquier usuario de Internet interesado en la materia, pero también pretende ser una herramienta que ayude a la docencia de las asignaturas de tratamiento digital de imagen (roles de profesores-alumnos). Por todo ello necesitamos saber que hay desarrollado al respecto en Internet, para no caer en los defectos y aprovechar las virtudes de los cursos existentes.

Analizaremos en detalle los cursos más similares, en cuanto a materia y a elementos interactivos, a los cursos que nosotros pretendemos crear con nuestra aplicación.

Antes de comenzar el análisis, debemos sentar las bases de lo que se define como una buena herramienta de aprendizaje basada en web, o lo que es lo mismo, del e-learning.

2.1.1. Introducción al e-learning (aprendizaje basado en web)

• *Definición de e-learning:*

El e-learning se define como el desarrollo del proceso de formación a distancia (reglada o no reglada), basado en el uso de las tecnologías de la información y las telecomunicaciones, que posibilitan un aprendizaje interactivo, flexible y accesible, a cualquier receptor potencial. Se entiende por teleeducación "una enseñanza a distancia, abierta, flexible e interactiva basada en el uso de las nuevas tecnologías de la información de de la comunicación, y de las comunicaciones, y sobre todo aprovechando los medios que ofrece la red Internet [*Definición de la Dirección General de telecomunicaciones de Teleeducación*].

En términos menos técnicos, definimos el e-learning como la enseñanza a distancia caracterizada por una separación física entre profesorado y alumnado (sin excluir encuentros físicos puntuales), entre los que predomina una comunicación de doble vía asíncrona donde se usa preferentemente Internet como medio de comunicación y de distribución del conocimiento, de tal manera que el alumno es el centro de una formación independiente y flexible, al tener que gestionar su propio aprendizaje, generalmente con ayuda de tutores externos. [*Definición en la Wikipedia: http://es.wikipedia.org/wiki/E-learning*].

• *Una solución e-learning está conformada por tres elementos fundamentales:* Plataforma de teleformación, Contenidos y Herramientas comunicativas.

- *La plataforma de teleformación (o LMS-Learning Management System):* es el elemento de hardware o software diseñado para automatizar y gestionar el desarrollo de actividades formativas a distancia (es el soporte tecnológico necesario para ofrecer la

educación a distancia). Suelen encargarse de registrar y almacenar datos de los usuarios, ofrecer herramientas de comunicación al servicio de los participantes en los cursos, etc. Las más utilizadas actualmente son WebCT (privada) [http://www.webct.com/] y Moodle (código libre) [http://moodle.org/].

- *Contenidos:* la calidad y cantidad de los contenidos es un elemento indispensable en un buen curso formativo. Deben presentar una estructura adecuada para su correcta asimilación. Además, los contenidos deben ser independientes de la plataforma en la que estén publicados, de tal forma que se garantice la Accesibilidad (independiente de la plataforma en la que estén los contenidos), Interoperabilidad (el contenido puede ser usado en diferentes plataformas), Reusabilidad (los contenidos puedes ser usados una y otra vez en diferentes programas educativos) y Durabilidad (el contenido podrá utilizarse sin importar los cambios en la tecnología en la cual se elaboró).

- *Herramientas comunicativas:* son aquellas que permiten la interacción entre los diferentes agentes del proceso de enseñanza-aprendizaje. Pueden ser herramientas síncronas: como teléfono, chat, webcam, videoconferencia, pizarra electrónica, etc. Las herramientas asíncronas propician más el e-learning entendido como "anytime,anywhere" y van desde los foros de debate, grupos de noticias, correo electrónico a los más actuales blogs o las Wiki.

• <u>*Características de la formación basada en web:*</u>

- Las principales características distintivas de la formación en red son:

· Aprendizaje mediado por ordenador. Uso de navegadores web para acceder a la información.

· Conexión profesor-alumno separados por el espacio y el tiempo.

· Formación interactiva: para ello se utilizan diferentes herramientas de comunicación tanto síncronas (tlf, chat…), como asíncronas (foros, e-mail…).

· Materiales digitales: Hipertextual-hipermedia y contenidos multimedia.

· Almacenaje, mantenimiento y administración de los materiales sobre un servidor web.

· Aprendizajde flexible. Aprendizaje individualizado y también colaborativo.

CARACTERÍSTICAS FORMACION BASADA EN WEB versus CARACTERÍSTICAS FORMACIÓN PRESENCIAL TRADICIONAL	
Características formación basada en la red	**Características formación presencial**
- Permite que los estudiantes vayan a su propio ritmo de aprendizaje. - Es "formación en el momento en que se necesita" (*just-in-time training*). - Puede utilizarse en el lugar de trabajo y en el tiempo disponible por parte del estudiante. -Es flexible. - El conocimiento es un proceso activo de construcción - Con una sola aplicación puede atenderse a un mayor número de estudiantes. - Tiende a realizarse de forma individual, sin que ellos signifique la renuncia a la realización de propuestas colaborativas.	-Parte de una base de conocimiento, y el estudiante debe ajustarse a ella. - Los profesores determinan cuándo y cómo los estudiantes recibirán los materiales formativos. - Puede preparase para desarrollarse en un tiempo y en un lugar. -Tiende a la rigidez temporal: se desarrolla en un tiempo fijo y en aulas específicas. - Parte de la base de que el sujeto recibe pasivamente el conocimiento para generar actitudes innovadoras, críticas e investigadoras. -Tiende a un modelo lineal de comunicación. Se desarrolla básicamente entre el profesor y el estudiante. -La enseñanza se desarrolla de forma preferentemente grupal.

- Permite la combinación de diferentes materiales (auditivos, visuales y audiovisuales) - No siempre disponemos de los recursos estructurales y organizativos para su puesta en funcionamiento. -Tenemos poca experiencia en su uso.	-Tiende a apoyarse en materiales impresos y en el profesor como fuente de presentación y estructuración de la información. - Disponemos de muchos recursos estructurales y organizativos para su puesta en funcionamiento. - Tenemos mucha experiencia en su utilización.

• *Ventajas e inconvenientes de la formación basada en web:*

- Las principales ventajas de la formación basada en web son:

· Bajo la idea de "formación en el momento en que se necesita" (*just in time* y *just for me)*: formación flexible que se puede aprender en cualquier momento y en cualquier lugar (deslocalización del conocimiento), y al ritmo que cada alumno necesite (autonomía del estudiante). Esto, por supuesto, ahorra costos y desplazamientos.

· Pone a disposición de los alumnos un amplio y esquematizado volumen de información, que puede ser actualizado en cualquier momento y reutilizado en diferentes cursos (actualización y reutilización del contenido).

· Favorece la interactividad con la información, con el profesor y entre los alumnos (ya no son sujetos pasivos), favoreciendo la formación grupal y colaborativa. Para ello se ofrecen diferentes herramientas de comunicación síncronas y asíncronas para los estudiantes y para los profesores.

· Permite que se pueda quedar registrado en los servidores la actividad realizada por los estudiantes, pudiendo realizar seguimientos individualizados y obtención de estadísticas de relevancia de contenidos, etc.

- Los inconvenientes mas destacados referidos a este tipo de formación son:

· Requiere de más inversión de tiempo por parte del profesor, y más trabajo en general que la formación convencional (lo que en muchos casos se traduce en la baja calidad de los cursos ofrecidos vía e-learning en la actualidad, ofreciendo una mala imagen de este método de aprendizaje).

· Requiere que los estudiantes tengan habilidades para el aprendizaje autónomo, ya que impone soledad y ausencia de referencias físicas (autodisciplina del alumno).

· Puede suponer una disminución de la calidad de la formación si no se da una interactividad adecuada ente profesores o/y alumnos.

· Precisa un conocimiento tecnológico básico por parte de profesores y alumnos.

· Se encuentra con la resistencia al cambio del sistema tradicional, del que ya hay mucha experiencia en su utilización.

· Depende de una buena conexión a Internet (conexión rápida para poder acceder fácilmente los contenidos multimedia y demás elementos interactivos).

· El soporte tecnológico de esta formación tiene los problemas típicos ligados a estos medios: problemas de seguridad y autentificación por parte del estudiante, suplantación de identidades, etc.

· Según los últimos datos sobre este tipo de formación, las cifras de fracaso en gestión de cursos a distancia ronda el 80% y se producen más de un 60% de abandonos de los cursos por parte de los estudiantes.

• *Recomendaciones de una buena web basada en el concepto de e-learning:*

- El *contenido* debe ser amplio, claro y *estructurado*. Debe ir acompañado de elementos *multimedia* que ayuden a su comprensión (imágenes, videos, audio). Evitar la linealidad (que provoca monotonía).

- Deben existir elementos que fomenten el aprendizaje requiriendo la actuación del usuario y permitiéndole evaluar los conocimientos adquiridos: ejercicios, applets con los que experimentar, autoevaluaciones, etc.

- Introducir herramientas de comunicación que fomenten la *interactividad* entre profesores-alumnos: foros, e-mail, blogs, grupos de noticias, etc.

- El *interfaz debe ser el adecuado*, siendo lo suficientemente intuitivo y estructurado para facilitar el aprendizaje del contenido de manera rápida y sencilla (debe permitir acceder rápidamente a partes concretas del contenido).

2.2. CURSOS DE TRATAMIENTO DIGITAL DE IMAGEN EN INTERNET

Para realizar una panorámica sobre los cursos de tratamiento digital de imagen que existen en la actualidad en Internet, vamos a reproducir lo que haría, por norma general, cualquier persona que desea aprender algo por Internet. Dicha persona suele dirigirse en primer lugar a un buscador (Google, Terra, Yahoo, etc.) e introducir las palabras clave que definen aquello que busca.

Partiendo de esa base, vamos a buscar "curso interactivo tratamiento digital de imagen" en uno de los principales buscadores de Internet (Google), para averiguar que se encontrará cualquier internauta interesado en la materia. Mostramos en la siguiente figura el resultado de esa búsqueda (podemos observar, que nuestro curso aparece muy bien situado en el ranking que hace Google sobre las webs relacionadas con cursos interactivos de tratamiento digital de imagen).

Figura 2.1: Búsqueda en Google de "Curso interactivo Tratamiento Digital de Imagen" con posición de IMAGine.

A continuación, analizaremos el resto de cursos, con características más o menos similares al nuestro, que hemos encontrado al realizar esta búsqueda o búsquedas similares (tanto en castellano como en inglés).

Palabras clave en español por las que hemos realizado las búsquedas han sido: (Cursos interactivos / Tutorial / Curso on-line) Tratamiento Digital Imágenes, Procesado de Imágenes, Restauración Imágenes. Applets tratamiento imágenes, Transformadas Imágenes. En inglés, las palabras clave utilizadas han sido: (course interactive/course/tutorial) image processing, image processing learning.

Analizaremos principalmente cinco aspectos: *estética* (apariencia de la web), *visualización de la web con distintos navegadores y resoluciones de pantalla*, *navegación* (facilidad/dificultad en la navegación), *contenido teórico* y *contenido interactivo*.

Puntuaremos cada uno de estos aspectos con los siguientes calificativos según la impresión (de peor a mejor) que nos han merecido: muy mal, mal, regular, bien o muy bien. Hemos de mencionar que esta clasificación no pretende ser objetiva, sino que pretende únicamente servir de referencia a la hora de hacer el diseño de nuestra propia web, ayudándonos a identificar de manera fácil y rápida los aspectos en los que IMAGine puede mejorar en relación a otras web similares ya existentes.

2.2.1. Curso interactivo de Trat. Digital de Imagen (Universidad de Málaga)

Este curso en general está bien, es muy fácil de utilizar y los cursos que contiene están bien diseñados, en cuanto a teoría y contenido interactivo, con una muy buena navegación y una estética atractiva. No obstante, los cuatro cursos de los que dispone, cubren solo un pequeño abanico de temas sobre tratamiento digital de imagen, quedándose mucha información en el tintero.

Los cuatro cursos tratan los siguientes temas:

- Curso *Aspectos básicos*: incluye nociones básicas de visión y colorimetría, así como de cuantificación de imágenes.

- Curso *Técnicas básicas*: incluye algunos aspectos relacionados con filtros, transformadas, restauración de imágenes mediante filtrado, y distorsiones geométricas

- Curso *Compresión y codificación de imágenes*: contiene información acerca de las principales técnicas de compresión y codificación de imágenes.

- Curso *Análisis de imágenes*: realiza una breve introducción sobre los temas de segmentación de imágenes, descriptores de contorno y procesado morfológico.

Título:	Tratamiento Digital de la Imagen. Curso Interactivo. Universidad de Málaga.
URL:	http://campusvirtual.uma.es/tdi/
Portada:	

Figura 2.2 : Curso Trat. Digital de Imágenes de la Universidad de Málaga.

A continuación pasamos ha hacer un análisis técnico pormenorizado de algunos aspectos de este curso:

• *Estética:* [Bien]

La estética del curso nos parece bastante atractiva (en formato de libro, con iconos, etc). En cuanto a la estética de la portada en sí, quizá se ve poco profesional (tiene la apariencia de una composición de imágenes unidas por un editor de imágenes), incluyendo los textos como parte de la imagen, dificultando así su lectura (puesto que no se ven bien definidos), y que además constituyen un problema en cuanto a *accesibilidad web* (personas con discapacidad no podrán leer ese texto porque está insertado como una imagen).

La web tiene un icono identificativo muy llamativo, que además se repite a lo largo del resto de las páginas, lo que nos ayuda a identificar el curso con ese símbolo.

La estética de los cursos (en formato de libro) se mantiene para todas las páginas de la web, manteniendo así la coherencia de la aplicación. Los tonos empleados además no dificultan la lectura ni resultan dañinos a la vista (se emplean tonos suaves).

• V*isualización de la web con distintos navegadores y resoluciones de pantalla:* [Muy Bien]

En la portada del curso, se informa de que la web está optimizada para cualquier navegador y resolución (detectando automáticamente el utilizado), pero que, no obstante, se recomienda utilizar una resolución y navegador en concreto. Además, ofrece la opción de seleccionar de forma manual el diseño de la web para una resolución dada.

• *Navegación:* [Muy Bien].

El curso se presenta en formato de libro electrónico, donde puedes navegar de manera lineal "pasando las páginas", pero también dispones de una serie de menús con accesos directos a contenidos concretos. En la parte central se muestra por lo general el contenido de los cursos.

La página ofrece una serie de menús siempre visibles que facilitan la navegación:

- El menú horizontal que aparece en la parte superior de la página:

º El "índice general" muestra un índice con todo el contenido teórico de la web con mayor o menor detalle dependiendo de la opción seleccionada.

º El resto de opciones ("aspectos básicos", "técnicas básicas", etc.) corresponde cada una a cada uno de los cursos que ofrece la web. Al pasar sobre ellas con el ratón se muestra un menú del propio curso, ofreciendo desde ellos accesos directos a partes concretas de los mismos.

- El menú de la derecha: ofrece información de interés general, como una guía de uso del curso (como funciona), las estadísticas de la web, requisitos mínimos para la visualización correcta de todos los elementos de la web, o direcciones de contacto. En la parte inferior hay un enlace a la portada del curso y un acceso directo al applet del

tema del curso por el cual estés navegando. Las flechas en la parte inferior son para la navegación por la web de forma lineal como si de un libro se tratase.

- El menú de la izquierda: cuando estamos en la portada, dicho menú contiene iconos con accesos directos a cada uno de los cursos que ofrece la web, pero cuando nos situamos dentro de unos de estos cursos en concreto, muestra el índice propio de dicho curso.

En general, la navegación por esta web es bastante buena, con menús siempre visibles tanto de la web en general, como del curso en particular, ofreciendo accesos directos y distintos modos de navegación (lineal o no lineal).

En cambio, echamos en falta algún índice que nos permita acceder de manera rápida a todos los applets del curso, sin tener que irnos metiendo en cada tema de cada curso para poder acceder a ellos.

• *Contenido teórico:* [Regular]

La teoría que abarcan los cuatro cursos de la web cubre un margen pequeño en lo que englobaría el tema de tratamiento digital de imagen.

No obstante, la teoría de los cursos no es demasiado densa, quizá hasta un poco escasa. Un punto a favor es que suele ir acompañada con imágenes explicativas. Otro de los defectos que encontramos es que quizá no está demasiado esquematizada (suele venir explicada en párrafos enteros, como si de un libro se tratase), y eso puede ser un punto en contra si se quiere impartir una clase apoyándonos en la web (digamos que la teoría se muestra de una manera óptima para que el usuario la lea de manera individual en su propio ordenador).

• *Contenidos interactivos:* [Regular]

En cuanto a los contenidos interactivos, el curso cuenta con applets (en general, uno por cada tema de un curso), que permiten experimentar con la teoría contenida en el mismo (el applet está accesible desde el propio menú del tema o desde el botón situado en el interior del menú de la derecha).

La mayoría de los applets cuentan con una breve introducción explicando su finalidad y en general son bastante sencillos de utilizar, incluso para un usuario inexperto.

En algunos casos, las imágenes de los applets son muy sencillas, por lo que los ejemplos solo quedan ilustrados de forma meramente didáctica, pero faltaría una perspectiva más realista, utilizando imágenes reales (con ello veríamos los efectos de estos algoritmos aplicados a problemas de imágenes reales).

Además, tenemos problemas para visualizar algunos de los applets del curso (no llegan a verse, como por ejemplo el de filtrado de mediana, filtrado inverso o filtrado de Wiener)… los autores de la web recomiendan bajarse el código del applet (ofrecen un enlace con el *archivo.java*) y ejecutarlos en un programa que compile y visualice estos applets en local. Esto, claro está, es un grave problema, puesto que muchos usuarios no dispondrán de los conocimientos necesarios para realizar este proceso, o en el peor de los casos, preferirán simplemente no molestarse en verlos.

Figura 2.3 : Curso Trat. Digital de Imágenes de la Universidad de Málaga → Applets

2.2.2. HIPR2: Image Processing Learning Resources (Universidad de Edimburgo)

Este curso es casi un manual de referencia en la materia de procesado de imagen, como indica el número de visitas (casi 400000 visitas desde Octubre del 2000) y el gran número de páginas que ofrecen un enlace a esta web.

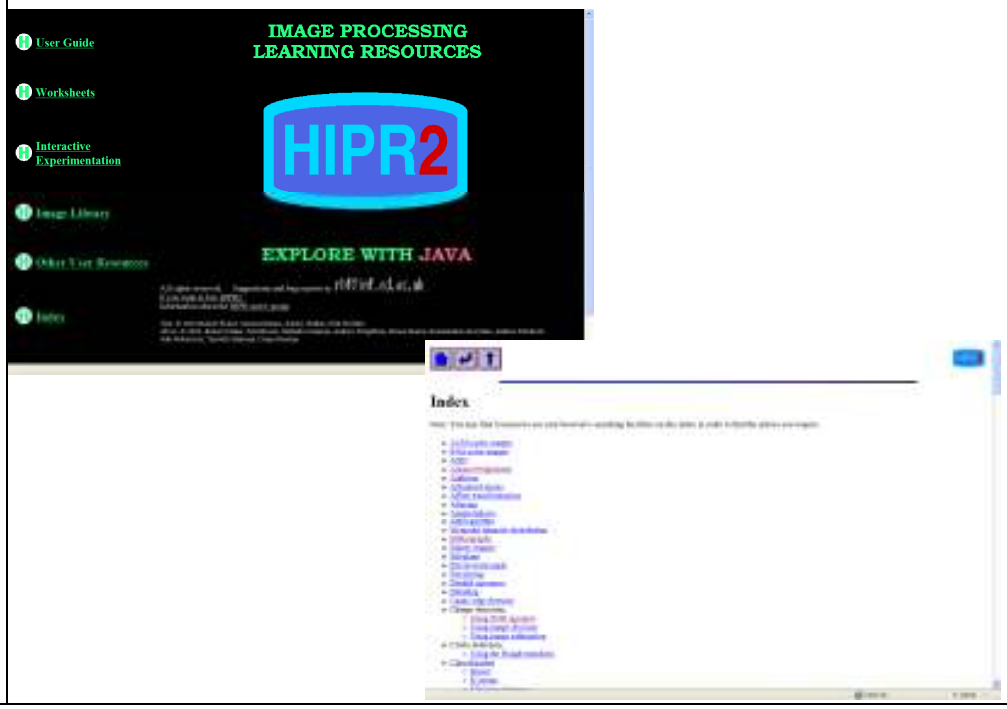
Título:	HIPR2: Image Processing Learning Resources explore with JAVA.
URL:	http://homepages.inf.ed.ac.uk/rbf/HIPR2/
Portada:	

Figura 2.4: Curso 'HIPR2: Image Processing Learning Resources' de la Universidad de Edimburgo.

Es un curso muy completo, puesto que abarca prácticamente la totalidad de temas relacionados con el tratamiento digital de imágenes, ofreciendo teoría (con ejercicios y referencias externas) y applets explicativos de cada apartado (muy elaborados en la mayoría de los casos). No obstante, tiene un gran defecto: la navegación se hace muy complicada, puesto que el índice general no está siempre visible. Esto conlleva que constantemente tengas que volver a la página principal para pasar de un apartado a otro

del temario. La estética tampoco está nada cuidada en esta web, resultando, entre otras cosas, muy monótona la lectura de los cursos y la navegación por la web.

Los temas (denominados *Worksheets* en la web) de tratamiento digital de imagen que trata este curso son principalmente: operaciones aritméticas, operaciones puntuales, operaciones geométricas, análisis de imágenes, procesado morfológico, filtros digitales, detectores de forma, transformadas de imágenes y síntesis de imágenes.

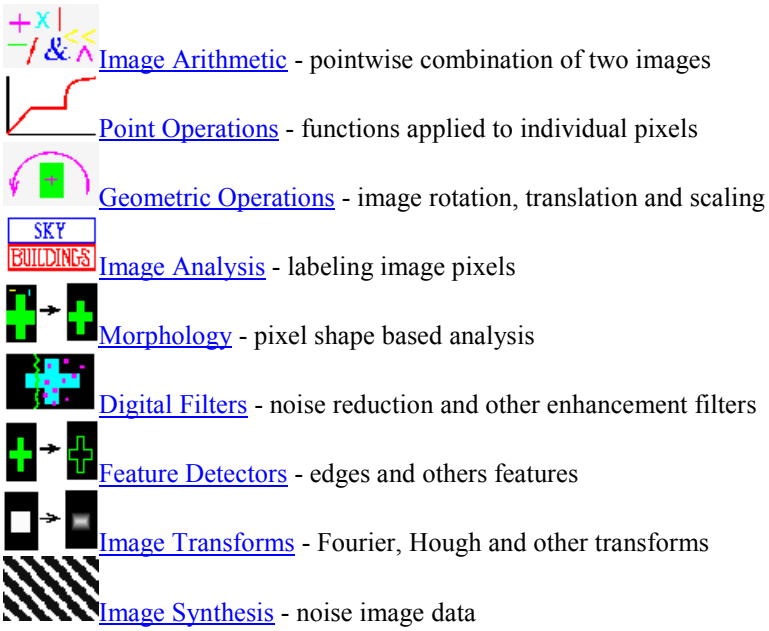


Figura 2.5: Curso HIPR2 → Índice Temas (Worksheets)

A continuación, pasamos a hacer un análisis más detallado sobre ciertos aspectos técnicos de la web:

• *Estética:* [Mal].

La web presenta un diseño sencillo y poco elaborado.

La portada tiene fondo negro y colores muy estridentes, presentando un alto contraste. A la hora de imprimir esta página con una impresora que no sea a color, apenas se distinguirán las letras del mismo, y si además ponemos que se imprima el fondo (tanto en impresora en grises como a color), el gasto de tinta será elevado, y si no, las letras que están en blanco no se verán.

El resto de páginas del curso (las que contienen la información propiamente dicha), presentan también un diseño muy sencillo, pero el tema de los colores mejora. Tienen fondo blanco y letras en negro, con los títulos resaltados en negrita y mas o menos siguiendo la misma estructura en todas las páginas. No obstante, los iconos y algunos dibujos esquemáticos contenidos en la teoría, siguen empleando colores muy estridentes rompiendo la armonía con el resto del contenido.

• *Visualización de la web con distintos navegadores y resoluciones de pantalla:* [Bien]

Al ser una página muy sencilla sin frames, ni estilos muy complicados que requieran elementos *div*, ni márgenes específicos, sino que se muestra siempre la información a pantalla completa, la visualización en cualquier navegador y con cualquier resolución

no afectará a la web. Además, es una página válida para HTML 4.0, lo que garantiza al menos que está bien construida (etiquetas correctamente anidadas, etc.), lo que hace que cualquier navegador la visualice correctamente.

• *Navegación:* [Mal].

En la portada, se ofrecen una serie de enlaces con información de carácter general (autores, guía de uso, requisitos del sistema para visualización correcta de la web, índice a los contenidos…).

Una vez visualizada la portada, el resto de la navegación por la web se hace prácticamente imposible, sobre todo en cuanto a la navegación por el contenido teórico de los cursos. Todas las páginas se muestran a tamaño completo y el índice desaparece, quedando solamente cuatro iconos para la navegación por la web que explicaremos más adelante.

Empecemos por el análisis de los índices: por un lado tenemos el índice por temas (*Worksheet*) y por otro el índice por conceptos (*Index*).

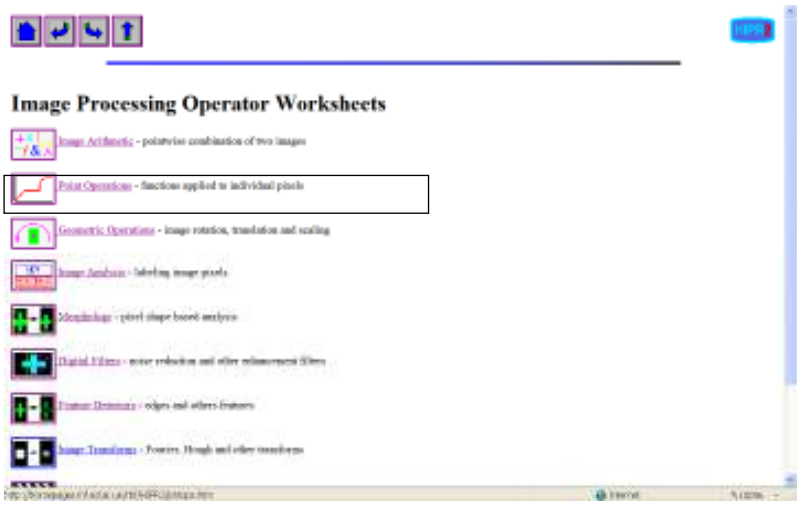
① El *índice por temas* ofrece una página con enlaces a cada tema. Dentro de la página del tema, se muestra en primer lugar un índice a los contenidos y posteriormente una breve descripción del tema. Dentro de cualquiera de los enlaces de los contenidos (cada uno de los contenidos se considera un "concepto" al cual se puede acceder de manera directa por el índice por conceptos que mencionábamos anteriormente) se muestra la teoría referente al concepto, un applet demostrativo, ejercicios y referencias bibliográficas.

En todas las páginas se muestra un menú de navegación que cuenta con cuatro iconos que permiten: volver a la portada, ir hacia el enlace anterior que estaba a su mismo nivel en el índice, ir hacia el enlace posterior que estaba a su mismo nivel en el índice, o subir al enlace que estaba en el nivel superior en el índice, volver a la portada, ir al concepto anterior, ir al concepto siguiente, subir de nivel.

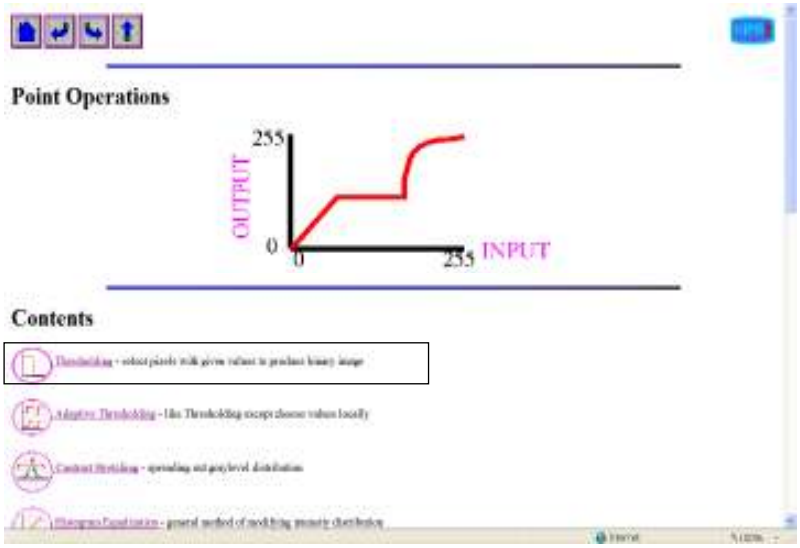
Vamos a mostrar un ejemplo de navegación en la siguiente figura:

---

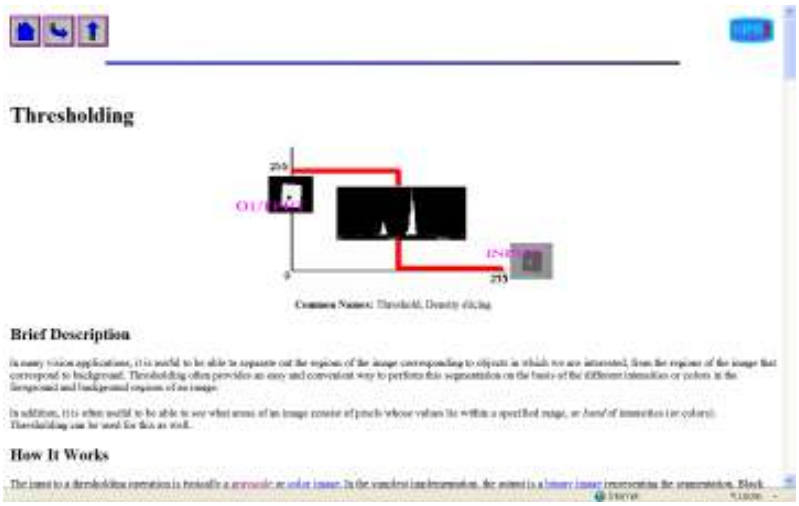
1) Seleccionamos el tema "*Point Operations*" dentro del índice de temas ("*Worksheets*"):



2) Seleccionamos el contenido “Tresholding” dentro del tema “Point Operations”:



3) Se muestra el concepto “Tresholding” con descripción acompañada por imágenes, applet explicativo, ejercicios y referencia bibliográfica:



* Si en el paso 3 pulsamos el icono de navegación con la flecha a la derecha nos conducirá al siguiente concepto que estaba junto a el en la página anterior→ “Adaptative Tresholding”. Si pinchamos en la flecha hacia arriba nos llevará a la imagen del paso 2 (página con la descripción del tema “Point Operations”), y así sucesivamente.

Figura 2.6: Curso HIPR2 → Navegación por el Índice de Temas

Vemos que la navegación se hace muy complicada al no estar siempre visible un índice con el esquema del curso, teniendo que volver continuamente al nivel superior o en el peor de los casos a la portada para empezar desde el principio (para poder situarnos claramente en dónde estamos situados dentro del curso).

② El *índice por conceptos* ofrece una página con enlaces a cada concepto. Se compone de todos los contenidos de todos los cursos descritos anteriormente (recordemos que los contenidos de cada curso estaban formados por conceptos). En el índice, estos conceptos se ordenan alfabéticamente, por lo que este índice nos servirá solamente si tenemos una duda concreta de algo muy específico.

En muchos casos, existen conceptos similares que ni siquiera están agrupados en uno solo (por ejemplo hay cuatro entradas distintas para píxel: *píxels*, *píxel format*, *píxel values* y *píxel connectivity*).

Una vez seleccionas uno de los conceptos, el índice desaparece y se muestra una página a tamaño completo con la información del concepto seleccionado. En la parte superior se sigue ofreciendo el menú de navegación con cuatro botones que comentábamos con anterioridad. En el caso de este índice, los únicos botones de este menú de navegación que tienen sentido son los que vuelven a la portada o suben de nivel, ya que nos permiten volver al índice general y seleccionar otro concepto que nos interese, porque dirigirnos al concepto de delante o de atrás del índice no tienen mucho sentido puesto que los conceptos que están en la posición anterior o posterior, al estar ordenados alfabéticamente, lo más probable es que no tengan ninguna relación entre si con el que estábamos mirando. En conclusión, continuamente tienes que estar redirigiéndote a la portada para acceder al índice de contenidos general y seleccionar el concepto que te interese estudiar en ese momento.

Un punto a favor, es que, una vez dentro del concepto, se muestra la teoría, un enlace al applet explicativo de ese concepto, ejercicios y referencias.

• *Contenido teórico:* [Muy bien]

La introducción a los cursos quizá no es muy extensa, pero en contrapartida, los conceptos que forman el contenido del mismo si vienen muy bien explicados, acompañados por imágenes y ejemplos.

Como además esta web abarca la mayoría de temas de tratamiento digital de imagen consideramos que la parte teórica está muy bien.

• *Contenidos interactivos:* [Muy Bien]

Este aparatado es sin duda la parte fuerte de este curso. Cada concepto cuenta con un applet demostrativo, y en la mayoría de los casos, suelen estar muy elaborados y resultan muy esclarecedores para ilustrar la teoría expuesta.

Cada uno de ellos cuenta con una introducción de la finalidad con la que han sido diseñados y el modo de uso, lo que facilita a cualquier usuario su manejo. Además, se proporciona un enlace al código fuente del applet e información en el mismo applet del tiempo que ha tardado en realizar el procesado de la imagen.

La imagen que se muestra en los applets se puede modificar de dos maneras: o bien poniendo la URL de una imagen o bien poniendo el nombre de algunas de las que hay cargadas en el servidor del propio curso (en la portada del curso hay un enlace donde muestra los nombres y apariencia de las imágenes que tienen disponibles). Este aspecto se podría mejorar, dando una opción que nos permitiese cargar una imagen, de entre las disponibles, de una manera más sencilla e intuitiva.

Como otro punto en contra, diremos que por ejemplo, en el caso del tema de procesado morfológico, al definir como conceptos distintos la erosión, la dilatación, la apertura y el cierre, hay un applet distinto para cada uno de ellos, y quizá en casos así, sería mas interesante contar con un solo applet que permitiese realizar las cuatro operaciones (eso nos permitiría apreciar mejor los efectos de cada uno y las diferencias

entre sí). Pero sin duda esto es una consecuencia de la estructuración de la web en conceptos distintos, y no en si de los applets, que como comentábamos, están muy logrados.

En la siguiente figura mostramos algunos de ellos:

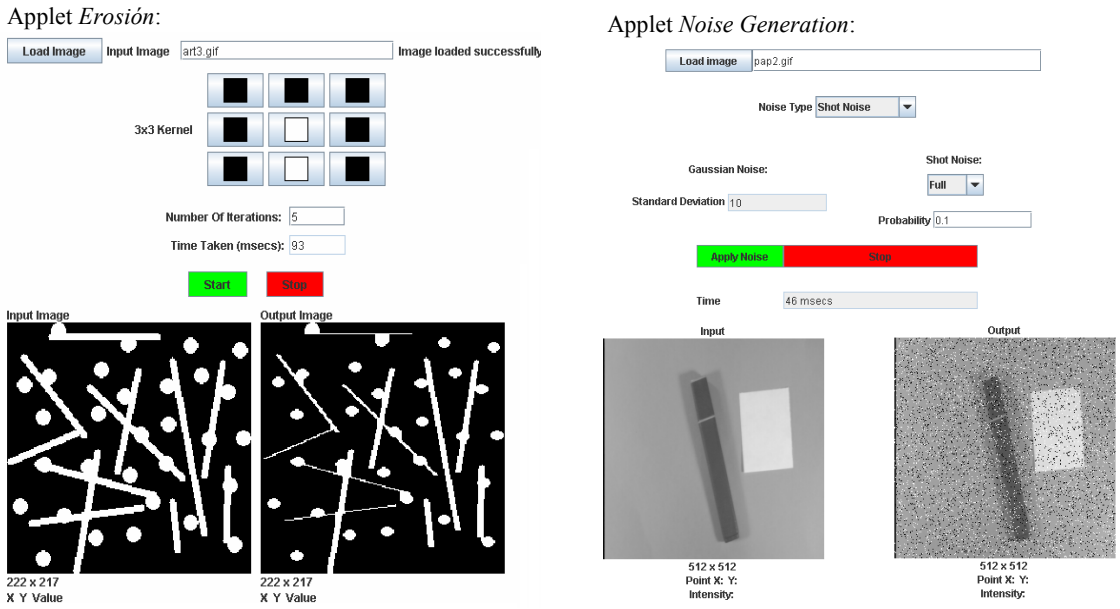


Figura 2.7 : Curso HIPR2 → Applets

2.2.3. Técnicas de Procesado de Imagen (Universidad de la Coruña)

Este curso está bien, puesto que incluye teoría y applets demostrativos en la mayoría de los apartados, pero quizá es demasiado sencillo (no cuenta con portada ni información adicional), un poco escueto (se basa principalmente en el tema del filtrado) y muy lineal (la única navegación posible es a través del índice o leyendo todos los apartados uno tras otro).

Los temas de tratamiento digital de imagen que trata son los siguientes (haciendo especial hincapié en el tema del filtrado):

- *Introducción al procesado de imagen*: incluye una breve introducción al tema del tratamiento de imagen.

- *Transformadas de intensidad*: especial mención al histograma y a diversas funciones predefinidas de transformación (negativo, umbral…).

- *Filtrado Espacial*: convolución, suavizado, realce, detección de bordes, eliminación de ruido y dilatación/erosión.

- *Filtrado Frecuencial:* introducción a la transformada de Fourier, filtros en frecuencia, filtrado en frecuencia y Experimento del Oppenheim.

Título:	Técnicas de Procesado de Imagen. Universidad de la Coruña.
URL:	http://www.des.udc.es/~adriana/TercerCiclo/CursoImagen/curso/web/Indice.html
Portada:	

Figura 2.8 : Curso Trat. Digital de Imágenes de la Universidad de la Coruña.

A continuación pasamos ha hacer un análisis técnico algo más pormenorizado de algunos aspectos de este curso:

• *Estética:* [Regular].

La estética de este curso es muy sencilla pero no cae en el empleo de colores estridentes ni otros elementos que rompan la armonía de la web.

• *Visualización de la web con distintos navegadores y resoluciones de pantalla:* [Bien]

Al igual que ocurría con el curso HIPR2, al ser una página muy sencilla sin frames, ni estilos muy complicados que requieran el empleo de elementos estructurantes, sino que en ella se muestra siempre la información a pantalla completa, la visualización en cualquier navegador y con cualquier resolución no afectará prácticamente a la web.

• *Navegación:* [Regular].

La navegación en esta web es muy lineal. La portada en sí es un índice con los contenidos de la web (que es en si un curso). Una vez dentro del contenido teórico del propio curso, disponemos del símbolo (representado por el icono del curso) que nos redirige al índice. Además tenemos unas flechas hacia delante y hacia atrás que nos vas a permitir ir a la página siguiente o a la anterior (la siguiente será el siguiente enlace que figuraba en el índice de la portada y la anterior la correspondiente al enlace anterior).

Por tanto, no tenemos accesos directos a los contenidos interactivos del curso (a los applets), y para dirigirnos a ellos tendremos que navegar por la teoría y averiguar en que páginas están insertados.

Además, el índice general no se muestra durante toda la navegación, teniendo que volver a él cada vez que se quiera tener una visión de conjunto sobre donde nos encontramos en cada momento.

• *Contenido teórico:* [Regular].

El contenido teórico de esta web está bien, pero abarca pocos temas de tratamiento digital de imagen, centrándose sobre todo en el filtrado.

Aparte de los propios applets, el contenido teórico apenas usa imágenes para apoyar las explicaciones, centrándose en la mayoría de los casos simplemente en mostrar las fórmulas. Un curso de procesado de imágenes debería emplear sobre todo imágenes de ejemplo para que se vea reflejado lo que se quiere explicar.

La forma de introducir la teoría (en formato de documento, sin esquematizarla por puntos, sino que más bien está orientada a que un usuario la lea de forma individual), imposibilita la utilización de la web con finalidad docente (es decir, visualizar la web con un proyector en una clase e impartir dicha clase apoyándose en ella), exceptuando los applets que si que se podrían utilizar como recurso docente en ese aspecto.

• *Contenido interactivo:* [Muy bien].

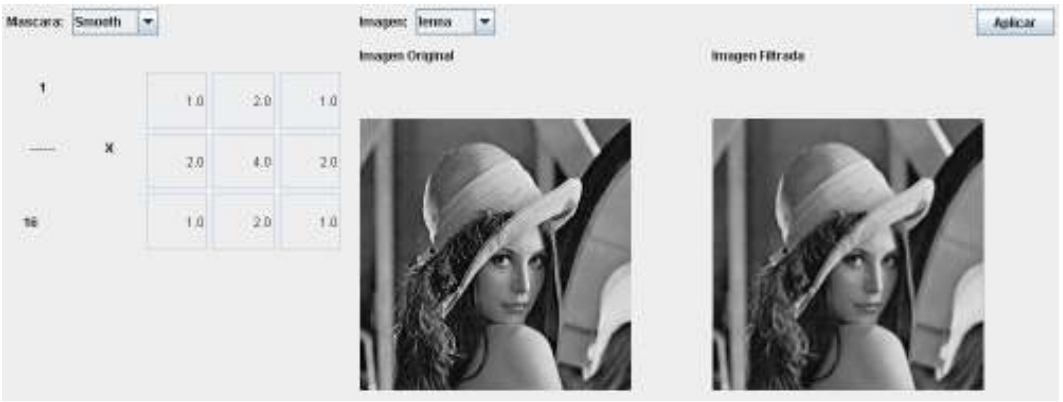
Los applets de este curso son applets muy completos a la par que sencillos y fáciles de usar. Muchos de ellos nos permiten hacer operaciones similares en un mismo applet para apreciar las diferencias entre ellas. También nos permiten variar algunos de los parámetros de configuración (tamaño del filtro, media del ruido…) para poder observar los efectos en cada caso.

Al estar insertados dentro de la propia teoría, no es necesario hacer una introducción sobre su finalidad, pero no obstante, en el propio applet suelen mostrar una breve explicación de cada operador o de cada técnica que se selecciona.

Las imágenes se pueden seleccionar de un desplegable donde viene el nombre. En este aspecto, se podría mejorar la carga de imágenes permitiéndonos visualizarlas en pequeño antes de cargarlas.

En la siguiente figura se muestran algunos de estos applets:

Applet *Filtrado Espacial: Suavizado*



Applet *Transformadas de intensidad: Funciones*

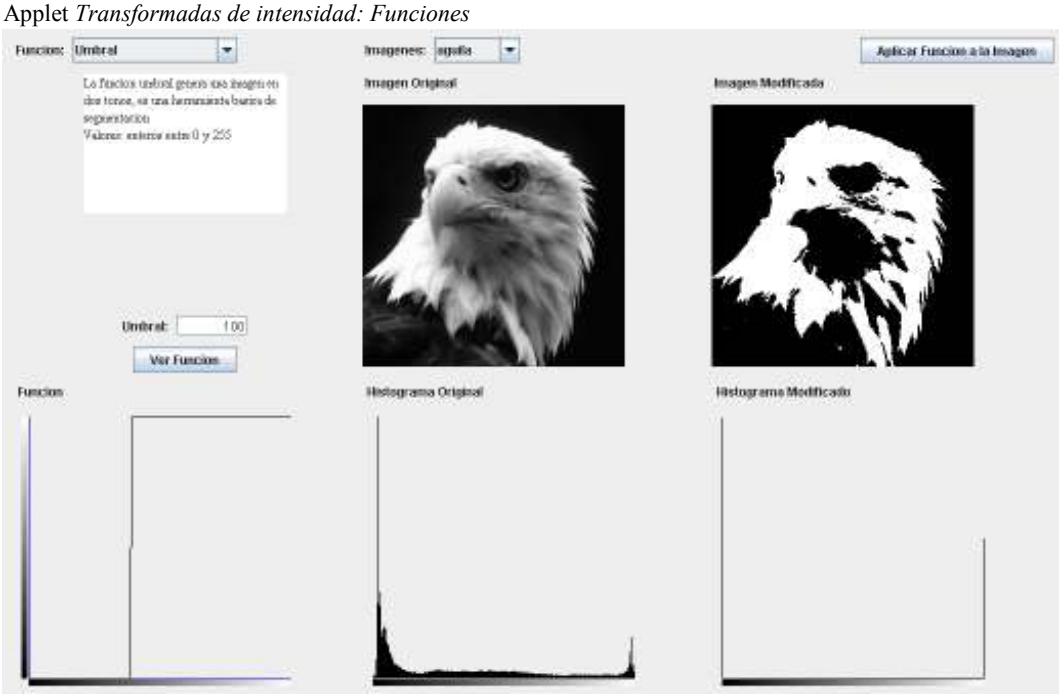


Figura 2.9 : Curso Trat. Digital de Imágenes de la Universidad de la Coruña → Applets

2.2.4. Otros cursos

Los cursos que hemos analizado hasta el momento, son cursos interactivos de tratamiento digital de imagen, es decir, incluyen teoría y applets. Nuestra aplicación (IMAGine) se pretende diseñar en base a este concepto.

El resto de cursos que hemos encontrado en la web, no cumplían este requisito, ya que en la mayoría de los casos son cursos meramente teóricos, y en otros casos, son cursos que solo contienen applets de procesado digital de imagen.

Por tanto, respecto a estos cursos, simplemente los mencionaremos a continuación y daremos una breve descripción de los mismos, ya que no se ajustan completamente a lo que estábamos buscando.

• *FIP: Fundamentals of Image Processing* (Delf University of Technology) [inglés]

URL: http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html

Se trata de un curso teórico en inglés sobre procesado de imágenes. Es bastante completo en cuanto a teoría, ya que abarca bastantes temas de tratamiento digital de imagen.

La estética de la web es sencilla, simplemente con una división en tres frames para facilitar la navegación y empleando colores estándar en webs de tipo teórico (fondo blanco con letras en azul y negro).

La navegación está bastante lograda (la página se muestra dividida en tres frames, mostrando en uno de ellos en todo momento el índice general, en otro unos iconos para seguir una navegación lineal, y por último, en el central, se muestra el contenido teórico), lo que facilita enormemente el aprendizaje de la materia.

El contenido teórico es el punto fuerte de esta web, abarcando un amplio abanico de temas sobre tratamiento digital de imagen. Además, la teoría viene acompañada de múltiples imágenes explicativas, que aunque no suplen la carencia de applets y otros elementos interactivos, si que facilitan enormemente el aprendizaje.

• *DIP: Digital Image Processing with Khoros* (Campinas University) [inglés]

URL: http://www.dca.fee.unicamp.br/dipcourse/index.html

Curso teórico, con multitud de imágenes explicativas, ejemplos y ejercicios, sobre tratamiento digital de imagen. Incluye también prácticas de laboratorio y ejemplos con la herramienta Khoros.

• *Curso Teórico sobre Tratamiento Digital de Imágenes:* (Iowa University) [inglés]

URL: http://www.icaen.uiowa.edu/~dip/LECTURE/contents.html

Curso teórico muy completo sobre tratamiento digital de imágenes, abarcando prácticamente la totalidad de los temas. La teoría incluye multitud de imágenes explicativas.

• *Curso Teórico de Procesado de imágenes digitales* (Universidad de Sevilla)

URL: http://www.sav.us.es/formaciononline/asignaturas/asigpid/

Curso sobre Topología Digital. Especial interés el Tema 4 que trata sobre Procesado Morfológico. Curso muy escueto que apenas trata el tema del procesado de imagen.

• *Curso Teórico de Procesado de imágenes digitales* (Universidad de Oviedo)

URL: http://wellpath.uniovi.es//es/contenidos/seminario/tutorialpdi/html/index.htm

Es un curso meramente teórico que ofrece nociones básicas sobre imagen digital, formatos de imagen, procesado y análisis de imágenes, etc.

• *Applets sobre tratamiento digital de imagen:*

- Applets para realizar procesados de imágenes a partir de la url de la imagen:
http://axon.physik.uni-bremen.de/online_calc/

- Otra página con enlaces a applets sobre procesado de imágenes:
http://www.cs.rit.edu/~ncs/graphics.html#educationAp

• *Otras herramientas/aplicaciones sobre tratamiento digital de imagen:*

- Descarga de una aplicación sobre procesado de imagen para ejecutar en local muy completa. Te permite procesar tus propias imágenes realizando filtrados, operaciones sobre píxels (negativo, grises, suavizado…), variar la intensidad de una imagen a color por componentes, muestra los histogramas…

Además incluye Control ActiveX (pieza genérica de software similar a una Librería de Enlace Dinámica (DLL) pero desarrollada en Visual Basic que genera un archivo con extensión .ocx y te permite incluir funciones de procesado de imágenes en tus aplicaciones sin necesidad de conocer los algoritmos que las implementan).

La URL para la descarga de la aplicación es la que sigue:
http://upiicsa.robotica.googlepages.com/ImageProcessorSample.zip

Se puede ver un ejemplo del funcionamiento de la aplicación sobre videos en la siguiente dirección:
http://www.youtube.com/watch?v=c2tH4AqH7uc

En la siguiente figura podemos ver una imagen de la herramienta de ejemplo que proporcionan utilizando las librerías ActiveX creadas para procesado de imágenes.

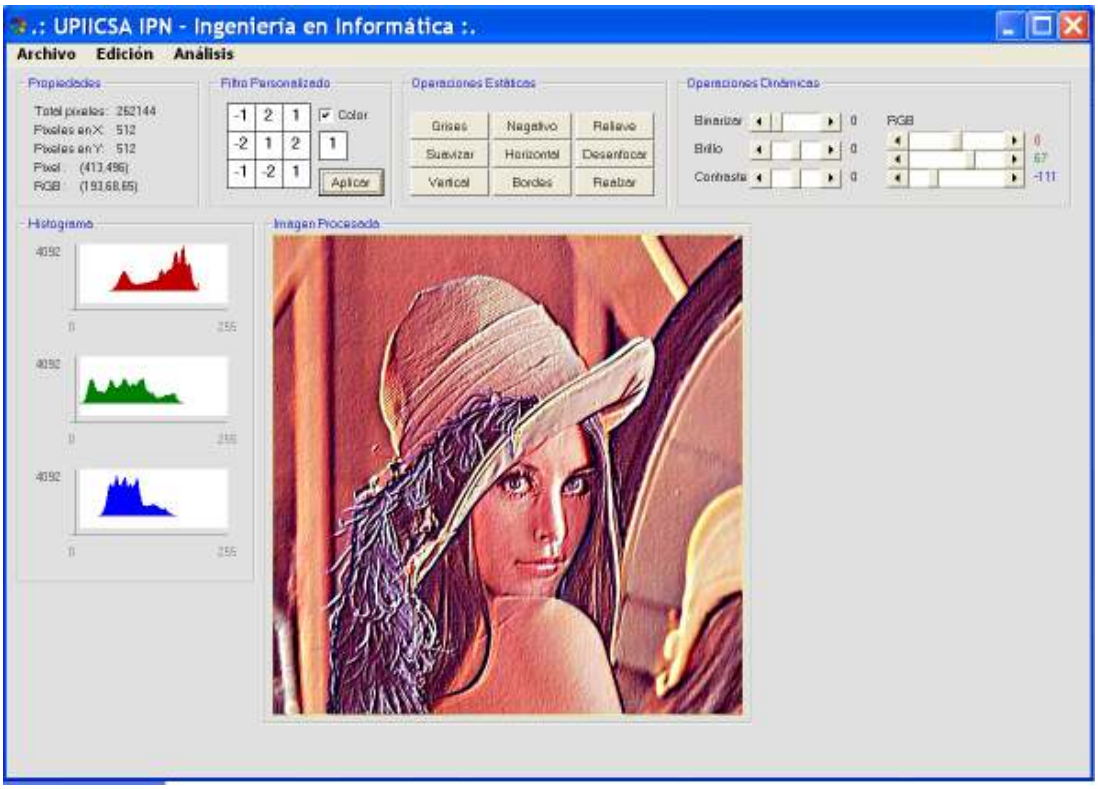


Figura 2.10: Aplicación procesado de Imágenes de UPIICSA

2.3. CONCLUSIONES

Antes de extraer las conclusiones, vamos a resumir la puntuación obtenida en el análisis anterior sobre los principales cursos similares a IMAGine existentes en la web. Con ello podremos identificar fortalezas y debilidades de cada uno de ellos en los aspectos analizados, para tenerlo en cuenta a la hora de diseñar nuestra propia web. Procuraremos no caer en los fallos cometidos en ellas e intentaremos igualar o incluso mejorar aquellos aspectos en los que ellas eran buenas.

Aspecto a analizar / Nombre del curso	**Estética**	**Visualización web con distintos navegadores y resoluciones de pantalla**	**Navegación**	**Contenido Teórico**	**Contenido Interactivo**
Curso interactivo de Trat. Digital de Imagen (Universidad de Málaga)	Bien	Muy Bien	Muy Bien	Regular	Regular
HIPR2: Image Processing Learning Resources (Edimburgo University)	Mal	Bien	Mal	Muy Bien	Muy Bien
Técnicas de Procesado de Imagen (Universidad de la Coruña)	Regular	Bien	Regular	Regular	Muy Bien
FIP: Fundamentals of Image Processing (Delf University of Technology)	Regular	Bien	Bien	Muy Bien	No tiene

Figura 2.11: Análisis cursos similares a IMAGine: cuadro resumen puntuaciones otorgadas.

Tras el análisis resumido en la tabla anterior [Figura 2.11.], obtenemos las siguientes conclusiones:

• Las webs que tienen una navegación complicada se hacen imposibles de visitar, desistiendo de visitarla tras navegar por las primeras pestañas (¿Dónde estaba el principio?, ¿en que parte de la teoría estoy en este momento?, etc.). → Es conveniente mostrar un índice en todo momento que nos permita tener una visión de conjunto y nos permita situarnos en cada momento.

• Las webs que contienen solo teoría se hacen demasiado densas y difíciles de seguir, dificultando el aprendizaje del temario que contienen. En contrapartida, aquellas webs que solo contienen elementos interactivos, omitiendo la teoría, presuponen unos conocimientos del usuario que éste no tiene porque poseer. El aprendizaje en ellas está basado en la experimentación con los elementos interactivos (pero en muchas ocasiones sin explicar siquiera el objetivo del elemento interactivo en concreto, sin decir que pretenden demostrar o experimentar con él). → Lo ideal sería una web que alternase teoría y elementos interactivos (son más dinámicas de visitar y facilitan el aprendizaje).

• Es conveniente facilitar enlaces directos a los contenidos interactivos para aquellos usuarios que solo estén interesados en esa parte de la herramienta (A estos usuarios puede que la parte teórica no les interese: Ej. profesor que está dando una clase y quiere ayudarse de un applet para explicar cierta materia).

• El alumno debe tener una herramienta que le permita evaluar los conocimientos adquiridos a lo largo del curso → Introducir autoevaluaciones, cuestionarios, ejercicios, sugerir problemas que el usuario pueda resolver….

• Applets:

- En los applets, es recomendable escribir una breve introducción sobre el objetivo del applet y como se utiliza.

- El motivo para insertar un applet es el de introducir un elemento interactivo a la aplicación, para que el usuario pueda experimentar con las distintas opciones que este ofrezca, y no ser un elemento meramente teórico (para eso no haría falta un applet).

- Los applets que incluyen demasiadas opciones se hacen difíciles de utilizar por un usuario inexperto, es mejor incluir applets con pocas opciones, más sencillos de utilizar, que se centren únicamente en un aspecto del temario (y no un applet donde se pueda probar todo junto poniendo valores en múltiples pestañas, que ha simple vista intimida al usuario porque no sabe que valores poner en todas las casillas…). En cambio si es conveniente que el usuario pueda modificar ciertos parámetros de configuración, pero se deben establecer en esos casos unos valores por defecto.

* Los applets del curso HIPR2 serían un buen ejemplo a seguir, ya que cumplen todos los requisitos descritos en los puntos anteriores.

• Otras aplicaciones de procesado de imágenes:

- La herramienta diseñada por UPIICSA (Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas) es un aplicación muy completa y útil, pero en un curso on-line por web, nos interesa tener pequeñas aplicaciones (applets) que sirvan para apoyar determinados puntos de la teoría, y no una herramienta que lo abarque todo.

- No obstante, en el capítulo 6 de esta memoria (titulado 'Líneas futuras'), sugerimos que una posible ampliación de este proyecto sería crear una herramienta similar a la de UPIICSA, pero centrada en el tema de restauración automática de imágenes, o que reúna todos los elementos diseñados hasta ahora en IMAGine en una única aplicación (esta opción es menos interesante puesto que en cualquiera de los applets de IMAGine, mediante el menú desplegable, se pueden realizar prácticamente todas las principales funciones que ya hay implementadas).

3. TEORÍA SOBRE TRATAMIENTO DIGITAL DE IMÁGENES

3. TEORÍA SOBRE TRATAMIENTO DIGITAL DE IMÁGENES ____ 43

3.1. INTRODUCCIÓN AL TRATAMIENTO DIGITAL DE IMÁGENES ____ 43

3.2. RESTAURACIÓN DE IMÁGENES ____ 49

3.2.1. Introducción a la Restauración de Imágenes ____ 49
3.2.2. Técnicas de mejora: casos de restauración ____ 50
3.2.3. Técnicas de restauración: restauración basada en modelos ____ 55
3.2.4. Aplicaciones de la Restauración de Imágenes ____ 59

3. TEORÍA SOBRE TRATAMIENTO DIGITAL DE IMÁGENES

[*Nota: Este capítulo pretende ser una introducción teórica de las técnicas de tratamiento digital de imagen utilizadas en el proyecto. Por tanto, un lector familiarizado con el TDI puede acudir directamente al capítulo siguiente.*]

Todo lo relativo a imágenes digitales ha cobrado una enorme importancia en los últimos tiempos, debido a los avances tecnológicos y a la amplia aceptación que han tenido los dispositivos de captura de imágenes digitales (es habitual encontrar cámaras digitales, escáneres y ordenadores en cualquier casa).

Por otra parte, en la sociedad actual también tiene una enorme importancia el lenguaje de la imagen. Los mensajes de contenido visual se utilizan frecuentemente para transmitir todo tipo de información, desde se utilización en las señales de tráfico hasta en el formato ágil y certero de la publicidad.

Sin embargo esta proximidad de mensajes gráficos que pretenden captar nuestra atención, el manejo habitual de aparatos de captura o la obtención de imágenes a través de la red, no siempre obtienen el resultado deseado. Para ello suele ser necesario realizar un procesado de la imagen tomada. Es ahí donde intervienen las técnicas de tratamiento digital de imagen.

Son muchos los conceptos asociados al procesado de imágenes digitales, por lo que trataremos de realizar una introducción que nos permita sentar las bases principales y realizar un recorrido por las herramientas más habituales.

3.1. INTRODUCCIÓN AL TRATAMIENTO DIGITAL DE IMÁGENES

El tratamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad, facilitar la búsqueda de información en ellas, alterarlas con cualquier fin concreto, etc.

Según González & Woods [*Ver referencia* [Bib. 3.1] *en Cap.10 Bibliografía*], "el objetivo de estas técnicas es procesar una imagen de tal modo que la resultante sea más adecuada que la imagen original para una aplicación específica". El término *específica* es importante porque dependiendo del objetivo de la aplicación y el tipo de imágenes que trata nos interesará utilizar un método u otro que resalte aquello relevante para la finalidad de la aplicación.

Las principales técnicas de tratamiento digital de imágenes son:

- Procesado básico de imágenes (operaciones puntuales, operaciones espaciales con filtros y transformadas).
- Técnicas de restauración de imágenes.
- Segmentación de imágenes.
- Transformaciones morfológicas.

- Extracción de características en imágenes, descriptores de objetos y reconocimiento de objetos.

- Indexado, búsqueda y recuperación de imágenes.

- Visión estereoscópica.

Para entender su base teórica y su finalidad, es necesario un análisis detallado de cada una de ellas. Eso se sale fuera de las pretensiones de este proyecto, por lo tanto solo nos centraremos en explicar la técnica añadida en el presente proyecto a las ya existentes en versiones anteriores. El curso añadido en *IMAGine versión 2.0* hace referencia a las técnicas de "Restauración de Imágenes".

Explicar como funcionan las técnicas de Restauración de Imágenes no sería posible si antes no definimos una serie de conceptos básicos sobre procesado de imágenes:

• Imagen digital:

Con este término nos referiremos a una función de dos dimensiones f(x,y), donde x e y indican las coordenadas espaciales, y el valor de la función en el punto (x,y) es proporcional al brillo (nivel de gris o intensidad) de la imagen en ese punto. Una imagen digital es una función f(x,y) que ha sido discretizada en ambas coordenadas espaciales (x,y) y en el nivel de gris [por tanto, una imagen digital es una matriz bidimensional cuyos elementos se denominan píxeles].

Consideramos que el nivel de brillo mínimo equivale al negro, el máximo al blanco y los valores intermedios son distintos niveles de gris que varían del negro al blanco. La profundidad de color de una imagen hace referencia a este concepto, es decir, al número de niveles de gris (o de colores) que empleamos para representar la imagen. Comúnmente se emplean 8 bits (256 niveles) para representar una imagen [si la imagen es en grises tendrá 256 niveles de gris, mientras que si la imagen es a color dispondrá de 256 niveles para la componente R, 256 para la componente G y 256 niveles para la componente B].

Figura 3.1: Imagen cuantificada con distintos niveles de gris
[Img Izda.)8 bits-256 colores; Img Centro)4 bits-16 colores; Img Dcha.)1bit-2colores]

Las ventajas de la imagen digital frente a la analógica vienen dadas por las múltiples posibilidades de manipulación que nos ofrece. A una imagen digital se le puede cambiar el contraste, el brillo, el color, el tamaño; se puede combinar con otras imágenes; se puede duplicar, rotar; puede cuantificarse y puede transmitirse a miles de kilómetros de distancia en pocos segundos.

• Digitalización de imágenes:

En una imagen digital el número de *píxeles* que la contienen es finito y su valor está cuantificado. Por tanto, el proceso de pasar una imagen analógica a digital implica un proceso que en la mayoría de los casos viene acompañado de pérdidas:

1. Muestreo espacial
2. Cuantificación de las muestras (L bits)
3. Codificación de los valores

Por tanto, se podría decir que la digitalización consiste en la descomposición de la imagen en una matriz de M x M puntos [muestreo], donde cada punto tiene un valor proporcional a su nivel de gris. Dado que este valor puede ser cualquiera dentro de un rango continuo, es preciso dividir dicho rango en una serie de k intervalos [cuantificación], de forma que el nivel de gris de cada punto sea asignado a uno de los valores que representa dicho intervalo [codificación].

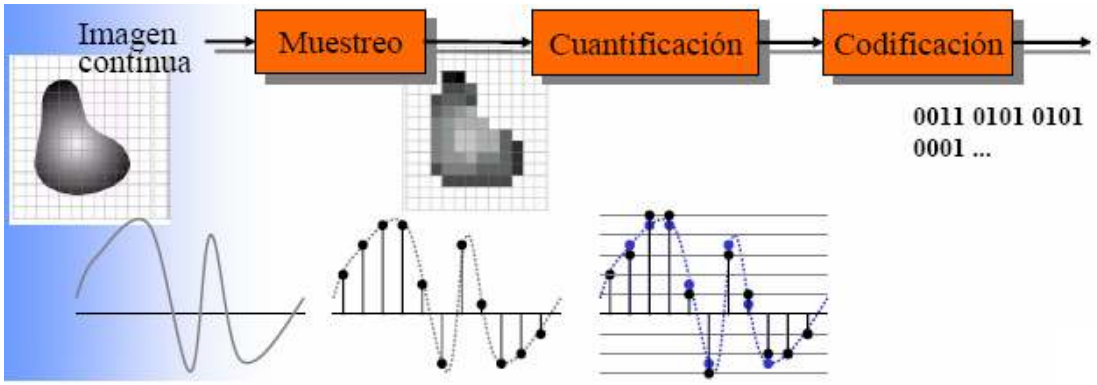


Figura 3.2: Diagrama Digitalización de Imágenes.

• Modelos de color:

Definen un color a partir de 3 componentes.

Modelo de color	Componentes	Uso
RGB	º R=Red (Rojo)→ 700 nm º G=Green (Verde)→ 546.1 nm º B=Blue (Azul) → 435.8 nm 	Es el modelo más utilizado para representar imágenes a color (es un modelo para fuentes de luz)

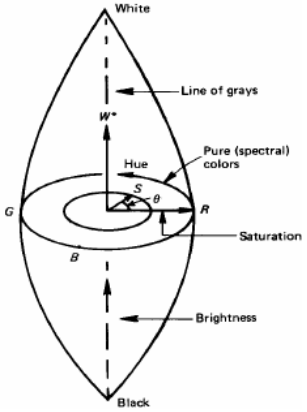
CMY	Se obtiene a partir del modelo RGB mediante una transformación lineal. º C=G+B → Cyan º M=R+B → Magenta º Y=R+G → Yellow 	Se utiliza para codificar colores en dispositivos de impresión
YIQ	Desacopla intensidad y color para compatibilidad con la televisión en blanco y negro. Se obtiene a partir del modelo RGB mediante una transformación lineal: $\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.513 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$	Utilizado en radiodifusión de TV del sistema NTSC (USA y Japón)
YUV	Desacopla intensidad y color para compatibilidad con la televisión en blanco y negro. Se obtiene a partir del modelo RGB mediante una transformación lineal: $Y = 0.30R + 0.59G + 0.11B$ $U = 0.493(B - Y)$ $V = 0.897(R - Y)$	Utilizado en radiodifusión de TV del sistema PAL. Usado en estándares de codificación digital JPEG y MPEG (más importancia a la Y)
HSI (o **HSV**)	º H: Hue (matiz) ⇨ Define el color (el ángulo determina el tinte) º S: Saturación ⇨ Cuán lejos está del blanco puro [va creciendo desde el centro (S=0, gris) hasta el borde (S=1, color puro)] º I: Intensidad Mantiene una relación no lineal con el modelo RGB $\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 1/\sqrt{6} & -2/\sqrt{6} & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$ $H = \tan^{-1}(V_2 / V_1)$ $S = [V_1^2 + V_2^2]^{1/2}$	Utilizado en TDI, pues todas las componentes tienen un significado perceptual. 

Figura 3.3: Modelos de color.

• **Histograma:**

El histograma de una imagen representa la frecuencia de aparición de un determinado nivel de gris en una imagen. Es una herramienta visual muy útil para apreciar la distribución de los niveles de gris de la imagen, el contraste que presenta y nos dará una idea del método más adecuado para manipularla.

En el eje de abscisas se mostrarán el rango de tonalidades (para una imagen en grises donde el color se representa con 8 bits, habrá 256 niveles de gris). En el eje de ordenadas aparecerán el número de píxeles que tienen esa tonalidad.

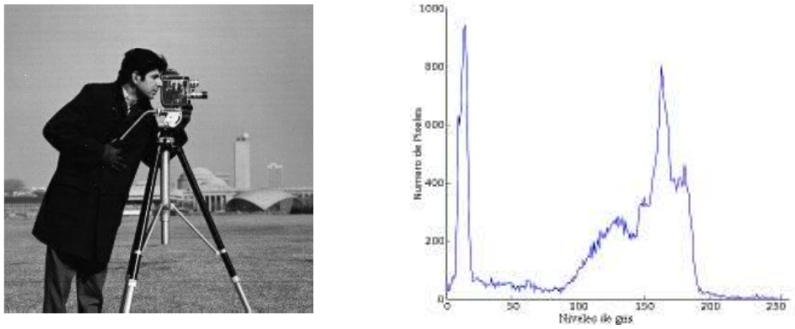


Figura 3.4: Histograma de una imagen.
Descripción: *la mayoría de la cantidad de píxeles se encuentran en el rango intermedio de tonalidades de gris (100-200), pero la elevación en los valores cercanos al cero corresponden a los píxeles con tonalidades más oscuras de la imagen (en este caso, los de la chaqueta del fotógrafo).*

En imágenes a color, se obtendría el histograma de cada una de las componentes por separado (si se pretenden manipular, para evitar el cambio de tonalidades al tratar por separado cada componente R, G y B, puede ser conveniente realizar un cambio de modelo de color como por ejemplo el HSI y manipular solamente la componente de intensidad).

• **Brillo y contraste de una imagen:**

El brillo de una imagen viene dado por la intensidad de sus píxeles. Una imagen cuya intensidad de sus píxeles en general sea baja, tendrá apariencia de poco brillo (más oscura). En cambio, una imagen dónde la mayoría de sus píxeles tengan tonalidades altas en media, será una imagen con mucho brillo (mas clara). Por tanto, modificar el brillo consiste en sumar o restar una cantidad fija a la intensidad de todos los píxeles de la imagen.

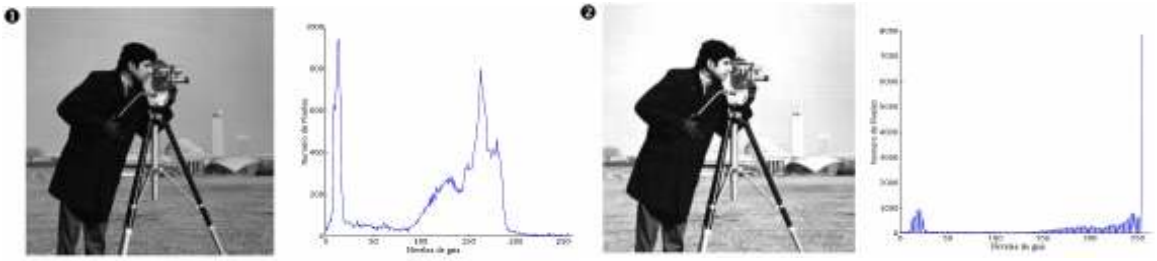


Figura 3.5: Brillo de una imagen.
Descripción: *en la imagen ❶ tenemos la imagen original y su histograma. En la imagen ❷ se ha sumado un 50% de brillo sobre la imagen* 1 *dando lugar a una imagen más clara, también con una distribución más homogénea de los píxeles (como se aprecia en el histograma), aunque con una mayor saturación en el nivel máximo (255) al haberse sumado intensidad a todos los píxeles.*

El contraste de una imagen viene dado por la diferencia entre las distintas tonalidades (claras y oscuras) de la misma. Permite diferenciar visualmente objetos plasmados en ella. Una modificación del contraste consiste en variar el rango dinámico de la imagen.

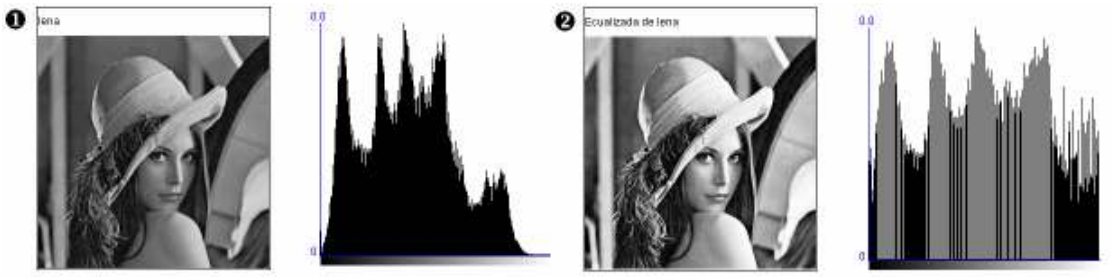


Figura 3.6: Contraste de una imagen.
Descripción: *en la imagen* ❶ *tenemos la imagen original y su histograma. En la imagen* ❷ *se ha ecualizado (igualado) el histograma sobre la imagen* 1, *dando lugar a una imagen con una distribución de tonalidades más uniforme en todo el rango.*

• **Transformada de Fourier:** (pequeña introducción)

La transformada de Fourier convierte coordenadas espaciales en frecuencias. Cualquier curva o superficie se puede expresar como la suma de un cierto número (quizá infinito) de curvas del seno y del coseno. En el dominio de Fourier (llamado el dominio de la frecuencia) la imagen se representa con los parámetros de las funciones del seno y del coseno. La transformada de Fourier es el mecanismo matemático para moverse dentro y fuera (transformada inversa de Fourier) del dominio de la frecuencia.

El dominio de la frecuencia se llama así porque los dos parámetros de una curva del seno son la amplitud y la frecuencia. El hecho de que una imagen se pueda convertir en una representación del dominio de la frecuencia implica que la imagen puede contener información de alta frecuencia o de baja frecuencia. Si el nivel gris de una cierta porción de la imagen cambia lentamente a través de las columnas, entonces sería representado en el dominio de la frecuencia como función del seno o del coseno teniendo una frecuencia baja. Una cierta cosa que cambia rápidamente, por ejemplo un borde, tendrá componentes de alta frecuencia.

Es por lo tanto posible construir los filtros que quitarán o realzarán ciertas frecuencias en la imagen, y éstas tendrán a veces un efecto reconstituyente.

En base a este principio, teniendo en cuenta que el ruido consiste principalmente en información de alta frecuencia, se podrá reducir utilizando un filtro que elimine estas componentes frecuenciales. Al eliminarlas, hay que asumir que también estaremos suavizando los bordes de la imagen (la información de los bordes también se localiza en las altas frecuencias).

En muchas ocasiones, es útil pasar una imagen al dominio de la frecuencia para obtener información de relevancia o realizar determinados procesados.

Otro concepto que debemos conocer es el de la convolución: cuando estemos hablando de una convolución en el dominio del tiempo, no nos estaremos refiriendo nada más que a una multiplicación en el dominio de la frecuencia.

3.2. RESTAURACIÓN DE IMÁGENES

3.2.1. Introducción a la Restauración de Imágenes

La restauración de imágenes es el arte y ciencia de mejorar la calidad de una imagen, en base a una medición absoluta. Implica generalmente algunos métodos de deshacer una distorsión que ha sido impuesta, por ejemplo un movimiento borroso o el grado de aspereza. Esto no se puede realizar de una manera perfecta, pero las mejoras son posibles en algunas circunstancias.

La restauración sería por tanto el proceso que trata de reconstruir o recuperar una imagen degradada basándose en algún tipo de conocimiento a priori sobre el proceso de degradación.

Distinguiremos entre dos tipos de técnicas para corregir una imagen distorsionada: aquellos realces efectuados sobre una imagen, orientados a conseguir una mejor visualización de la misma (mejora del color, del contraste, del brillo, etc.) y aquellos cuyo objetivo es eliminar imperfecciones de la imagen provocadas por una distorsión (mejorar la nitidez o el enfocado, eliminar ruido, etc.).

- Así las técnicas de restauración se orientan hacia la introducción de modelos de degradación, que luego se aplican en sentido inverso para recuperar la imagen original. Ejemplo: mejora de la nitidez de una imagen mediante la aplicación de una función que elimine el difuminado (es una técnica de restauración puesto que se corrige averiguando la función de degradación que la ocasionó).

Dentro de estas técnicas de restauración basada en modelos se incluirían:

· Distorsión por desenfoque.

· Distorsión por movimiento.

· Distorsión por imagen doble.

- En cambio, las técnicas de mejora son básicamente procedimientos heurísticos diseñados para manipular una imagen aprovechando los aspectos psicofísicos del sistema visual humano. Ejemplo: mejora del contraste de una imagen en base a la igualación del histograma (es una técnica de mejora puesto que se basa en que el aspecto de una imagen con un histograma más igualado es más agradable al observador).

Dentro de estas técnicas de mejora se incluirían:

· Mejora del contraste

· Cancelación de interferencias sinusoidales

· Eliminación de ruido

· Correcciones geométricas

· Restauración de color

· Restauración de secuencias de video.

3.2.2. Técnicas de mejora: casos de restauración

Hay imágenes que se han degradado debido a procesos heurísticos (por ej. ruido), o han sufrido un deterioro debido al paso del tiempo (por ej. deterioro del color o grietas). El objetivo de las técnicas de mejora es obtener una imagen más agradable visualmente que la imagen de partida. Para ellos hay una serie de técnicas que nos permiten modificar alguno de estos aspectos:

• *Mejora del contraste:*

Si realizamos una igualación del histograma de una imagen dada, seguramente la imagen resultante sea más agradable al ojo humano (porque su distribución de tonalidades estará más repartida por todo el rango de grises y nuestro ojo podrá percibir mejor los contrastes de la imagen).

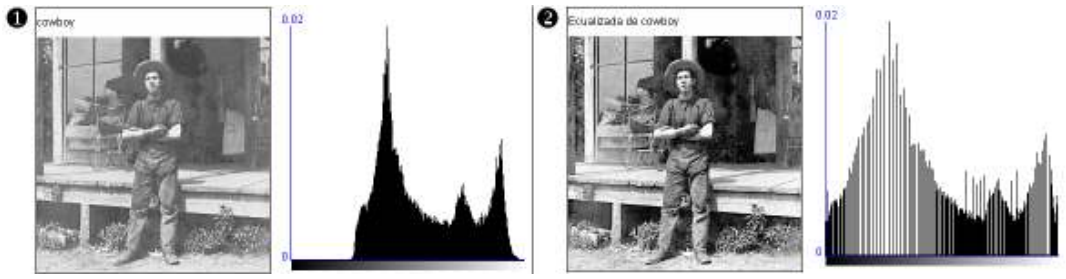


Figura 3.7: Ejemplo mejora del contraste.
Descripción: ❶*Imagen original*. ❷*Imagen ecualizada mediante una igualación del histograma.*

Esta afirmación no siempre es válida. Hay ocasiones en que aplicar una igualación del histograma no modifica favorablemente la calidad de la imagen final, como sucede con las imágenes de alto contraste. En esos casos, es preferible especificar un histograma particular para la imagen de salida con el fin de obtener los resultados deseados.

En imágenes a color, se debe modificar por separado el histograma de cada uno de los componentes de la imagen (R, G y B). Este también puede provocar efectos indeseados modificando los colores de la imagen final resultante. Puede ser conveniente realizar un cambio en el modelo de color, por ejemplo hacia el espacio HSI, y modificar solamente el histograma de la componente I que corresponde con la intensidad, así no variaremos la tonalidad del color, especificado en las otras dos componentes (la H-matiz y la S-saturación).

• *Cancelación de interferencias sinusoidales:*

Algunas imágenes sufren un patrón de interferencias que en ocasiones puede ser eliminado mejorando la calidad de la imagen final. Cuando la interferencia es aditiva y selectiva en frecuencias (solo se da en determinadas frecuencias) puede eliminarse por filtrado (eliminando las frecuencias afectadas por la interferencia). En estos casos la calidad de la imagen tras el filtrado mejora significativamente.

El procedimiento que se sigue en este tipo de casos es realizar la DFT de la imagen distorsionada para localizar las frecuencias en las que se produce la interferencia. Se aplica una máscara sobre esas frecuencias para eliminarlas y finalmente se realiza la DFT inversa sobre el resultado de aplicar la máscara para reconstruir la imagen sin las interferencias.

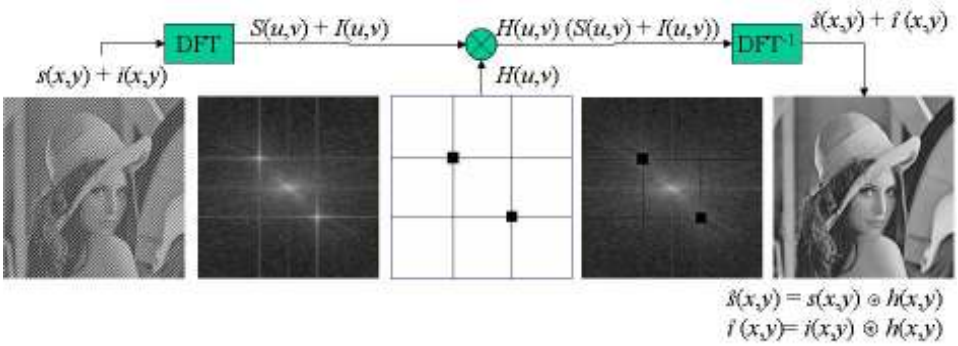


Figura 3.8: Modelo cancelación de interferencias sinusoidales.

• *Eliminación de ruido:*

Hay numerosas causas por las cuales una imagen puede manifestar ruido. Los aparatos eléctricos, tanto de captación de imágenes, transmisión o recuperación de las mismas pueden introducir ruido de naturaleza aleatoria en las imágenes.

El ruido se presenta en píxeles aislados (o casi) de color muy diferente a su entorno. Existen diversos tipos según su naturaleza estadística. Los más habituales son el ruido gaussiano (cuya función de distribución estadística tiene forma de campana) o el ruido impulsivo (cuya función de distribución estadística tiene forma rectangular).

La forma de combatir el ruido en las imágenes es mediante el filtrado. Según la naturaleza del ruido, será más eficiente un filtrado u otro, pero en todos los casos la imagen final tendrá sufrirá un suavizado en los bordes (el filtro realiza un promediado de los píxeles, en mayor o menor medida):

- Para la eliminación del ruido impulsivo es más eficiente el filtro de mediana. Este filtro, para obtener el valor de un píxel, ordena de menor a mayor todos los de su vecindad (se cogerán más o menos vecinos según el tamaño de la máscara que utilice el filtro) y asigna como valor del píxel que se quiere calcular el valor del píxel que se encuentra en la posición central de la lista ordenada realizada previamente. Así, un valor muy atípico respecto al de sus vecinos (el del ruido), nunca será elegido como valor de un píxel, y ni siquiera afectará al valor del píxel (en cambio, con el filtrado paso bajo, al realizar el promediado y el ruido corresponder a un valor atípico, seguirá afectando al valor del píxel resultante).

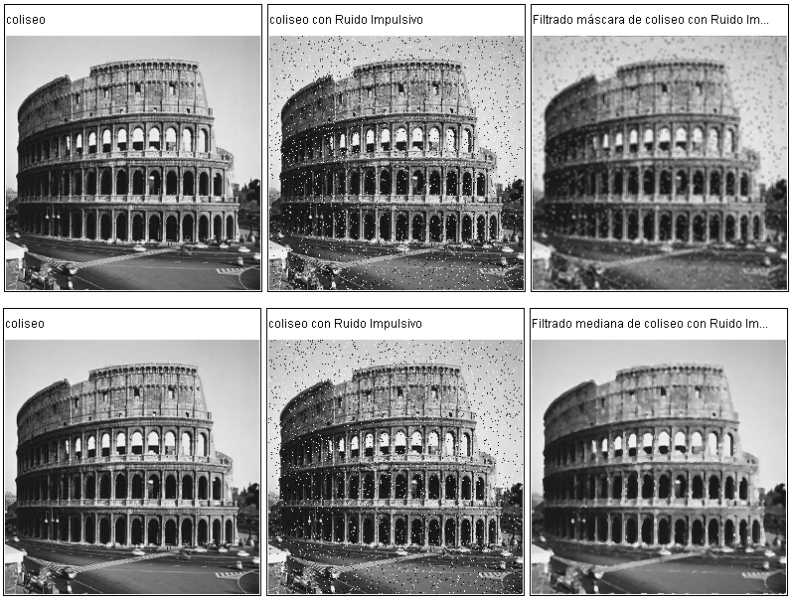


Figura 3.9: Eliminación de ruido impulsivo con filtro paso bajo y de mediana.

- Para la eliminación del ruido gaussiano es más eficiente el filtro paso bajo. El filtro paso bajo utiliza una máscara que, para obtener el valor de un píxel, coge todos los píxeles de la vecindad y los promedia en función de los parámetros del filtro (dando más peso a unos que a otros). El valor del promedio obtenido será asignado para ese píxel en la imagen final.

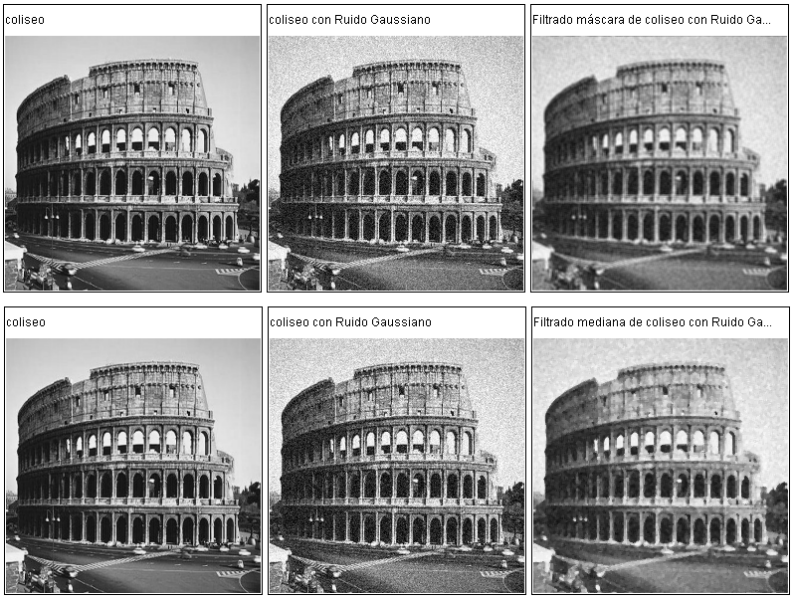


Figura 3.10: Eliminación de ruido gaussiano con filtro paso bajo y de mediana.

• *Correcciones geométricas:*

Las transformaciones geométricas implican un cambio de posición de un píxel a un nuevo lugar. Por tanto, son operaciones que permiten manipular la posición de la imagen en el eje de coordenadas, así como ajustar la escala y las distintas proyecciones de perspectiva.

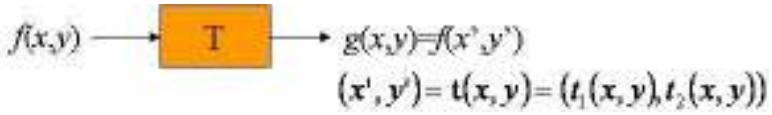


Figura 3.11: Modelo para la transformación geométrica de una imagen.
Descripción: *Dada una imagen f(x,y), si aplicamos una función de transformación T(x,y), obtendremos una imagen de salida g (x',y'), ya que T realiza una transformación sobre las variables x e y (que indican la posición de los píxeles en la imagen de entrada) convirtiéndolas en x' e y'.*

Con esta técnica, se tiende a distorsionar la calidad visual de la imagen. Dado un punto de coordenadas enteras (x,y), al aplicar la función de transformación para calcular la nueva posición de ese píxel, no se obtendrá unas coordenadas (x',y') que sean números enteros, y será necesario un proceso de interpolación. Toda interpolación disminuye la calidad de la imagen.

Para aminorar este problema, existen algoritmos de interpolación bastante eficaces:

· Algoritmo NEAREST (el vecino más cercano): asigna a la nueva casilla el valor más cercano con relación a la posición original dejando intacto el valor del píxel.

· Algoritmos BILINEAR y BICUBICO: asignan a la nueva casilla el valor más cercano con relación a la posición original, asignándole el valor resultado de realizar una interpolación bilinear y cúbica respectivamente, tomando en cuenta los valores de los píxeles vecinos.

Las distorsiones geométricas más comunes son las denominadas afines o lineales (aquellas que afectan a la escala y/o a la posición, incluyendo aumentos, rotaciones y traslaciones), pero también existen otras más llamativas que requieren un cambio del plano de proyección (cilíndrico, esférico o como una malla con determinada forma, como en el caso de las distorsiones *pin-cushion* y *barrel*).

Figura 3.12: Ejemplos distorsiones geométricas.

• *Restauración de color:*

El paso del tiempo afecta negativamente a los colores de las imágenes impresas en papel fotográfico. Muchas de ellas llegar a presentar un tinte casi monocromático. Dependiendo del estado de deterioro, en ocasiones es posible restaurar los colores originales de la imagen con resultados más que aceptables.

No solo el paso del tiempo afecta al color de las imágenes, en ocasiones, una mala captación, transmisión o recuperación de las mismas (quizá debido a las condiciones ambientales o a la calidad del objetivo, de la película, del aparato de transmisión o recuperación de imágenes, etc.), hace que ciertas tonalidades se vean afectadas.

El proceso de restauración del color en imágenes es una labor bastante artesanal, que requiere de un proceso individualizado para cada tipología de degradación, incluso casi para cada fotografía. Los resultados obtenidos casi nunca son perfectos, aunque si suelen mejorar considerablemente la calidad de las imágenes de partida (siempre y cuando se ha realizado un trabajo minucioso).

El primer paso en el proceso de restauración del color es digitalizar la imagen. Hay que tratar de averiguar la *función de decoloración* sufrida por la misma para poder aplicar su inversa restaurando así los colores originales. Cuando la pérdida de color se debe al paso del tiempo, hay funciones de decoloración ya estimadas que reproducen aproximadamente el efecto del tiempo en los colores según las tonalidades, y que nos pueden servir de gran utilidad.

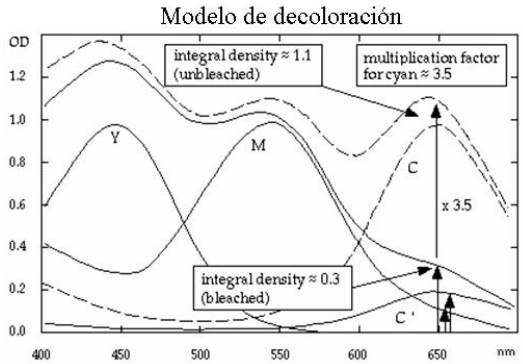


Figura 3.13: Modelo de decoloración típico de una imagen a color deteriorada por el paso del tiempo.

Por último, se debe aplicar la transformada inversa de la función de decoloración estimada. Sino se ha conseguido estimar exactamente (el proceso puede ser muy laborioso), seguramente se hayan detectado en el proceso las principales degradaciones sufridas (por ejemplo, cierta tonalidad anaranjada en toda la imagen), que se deberán corregir. Finalmente, si se desea, se puede pasar la imagen restaurada al formato analógico.

Figura 3.14: Ejemplos restauración imágenes a color.

• *Restauración de secuencias de video:*

El proceso de restauración de secuencias de video utiliza las principales técnicas de procesado temporal. Dichas técnicas suelen fundamentarse en la interpolación entre imágenes aledañas dentro de una misma escena. Para ello es necesario haber estimado previamente el vector de movimiento entre sucesivos fotogramas (queremos saber donde se sitúa un píxel de un fotograma en los fotogramas vecinos, para así obtener la información que nos falta).

Con esta información, podremos reconstruir los fragmentos deteriorados, corregir sacudidas y parpadeos, eliminar ruido e interferencias, etc.

Lo más habitual es requerir de técnicas de restauración de video para recuperar películas antiguas, que debido al paso del tiempo, tienen muy poca calidad y se hace difícil su visualización.

-El proceso de restauración de secuencias de video se divide en las siguientes etapas:

1) *Digitalización:* imprescindible para realizar el proceso de restauración por ordenador aplicando las nuevas técnicas digitales.

2) *Segmentación por planos:* proceso (manual normalmente) que requiere fragmentar la película en escenas y posteriormente en planos para determinar que imágenes deberían guardar cierta relación entre si para poder continuar con el proceso de restauración.

3) *Estimación y corrección de sacudidas:* las sacudidas se producen debido al distinto posicionamiento de sucesivos fotogramas dentro de un mismo plano (bien debido a movimientos de cámaras sostenidas manualmente durante la grabación, o por errores en el posicionamiento de los fotogramas durante la digitalización). Se corrige estimando el movimiento sufrido y corrigiéndolo mediante transformaciones afines (traslación, aumento, rotación).

4) *Corrección de parpadeo:* se debe a fluctuaciones de la luminosidad en las distintas imágenes que conforman una misma secuencia. Se corrige controlando el histograma de las imágenes de esa secuencia de forma que varíen en imágenes sucesivas de un modo más uniforme.

5) *Supresión de manchas, líneas y ruido granular:* se deben a alteraciones de la película en formato analógico por el paso del tiempo (adhesión de partículas, deterioro de la película, etc.). Se corrigen comparando fotogramas sucesivos dentro de un mismo plano. Primero se estima la función de movimiento entre fotogramas consecutivos y después se remplazan los píxeles dañados en un fotograma por los correspondientes en los fotogramas vecinos. Para la corrección de ruido se pueden emplear las técnicas habituales de filtrado.

6) *Realce de imagen, recorte y aumento:* se realzan los bordes de las imágenes para compensar los efectos de filtrados previos. Además, se suprimen marcos negros (si los hubiese fruto de la corrección de sacudidas) mediante recortes o aumentos.

3.2.3. Técnicas de restauración: restauración basada en modelos

Obtener la imagen perfecta dada la imagen degradada es la meta de la restauración, y no es generalmente posible. Por lo tanto deseamos mejorar la imagen tanto como sea posible.

Para ellos partimos de los llamados modelos de degradación, que nos permiten estimar la función de degradación a la que ha sido sometida la imagen original y que ha dado lugar a la imagen distorsionada que tenemos. Conocida dicha función, se calculará su inversa y se aplicará por convolución a la imagen distorsionada. Con ello pretendemos obtener la imagen original sin distorsión.

Definamos los conceptos principales de nuestro modelo: existe un operador (o un sistema) *h*, que junto a un término aditivo de ruido *r(x,y)*, actúa sobre una imagen de entrada *f(x,y)* para producir una imagen degradada *g(x,y)*.

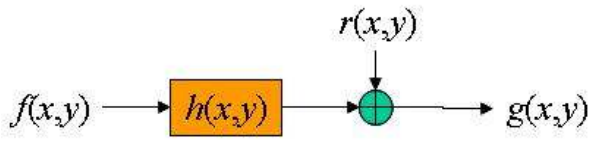


Figura 3.15: Modelo de degradación de imágenes.

Por tanto, la restauración de imágenes, en base a este modelo, podría entenderse como el proceso de obtener de forma aproximada *f(x,y)* a partir de *g(x,y)* conociendo una estimación de la degradación introducida por el operador *h(x,y)*. Se supone el conocimiento de *r(x,y)*, que será un ruido representado por una función de naturaleza estadística.

Al modelar así el problema, estamos asumiendo un sistema espacialmente invariante, donde la función de degradación (también conocida como PSF en inglés) es igual en todos los puntos de la imagen. Esto no siempre es así en la realidad, por eso, en los casos en los que el sistema no sea espacialmente invariante, se hará la reconstrucción en pequeñas regiones de la imagen asumiendo que en esa zona, el PSF es casi constante.

Lo más importante para restaurar la imagen es estimar correctamente el PSF (el *h* en nuestro modelo). Para ello, lo normal es partir de de una imagen original sin distorsión a la que le aplicamos un sabido PSF por convolución (es una forma de medir los resultados objetivamente, ya que disponemos de todos los datos).

Para seguir este procedimiento, nos solemos fijar en un borde supuestamente abrupto de la imagen, después suponemos una PSF Gaussiana con determinada desviación típica (que iremos variando) y se la aplicamos a la imagen sin distorsión. Si la imagen distorsionada obtenida tiene rasgos similares a la que nosotros queremos restaurar, ese será el PSF cuya inversa tendremos que aplicar.

También podemos proceder aplicando la inversa de varias PSF Gaussianas con distintas desviaciones típicas sobre la imagen distorsionada que queremos restaurar para ver con cual obtenemos mejores resultados de restauración.

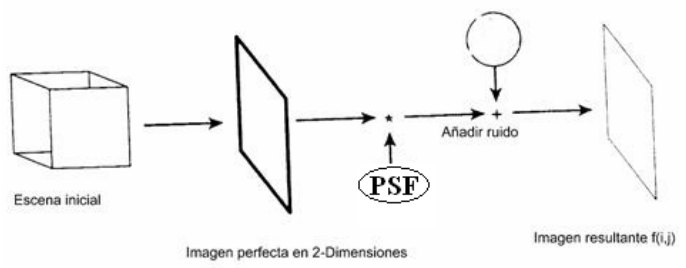


Figura 3.16: Modelo de degradación de imágenes. ¿Cómo obtener la función de degradación (PSF)?

- Existen <u>funciones de distorsión típicas</u>, como las que producen en las imágenes desenfoque, movimiento o un efecto de imagen doble.

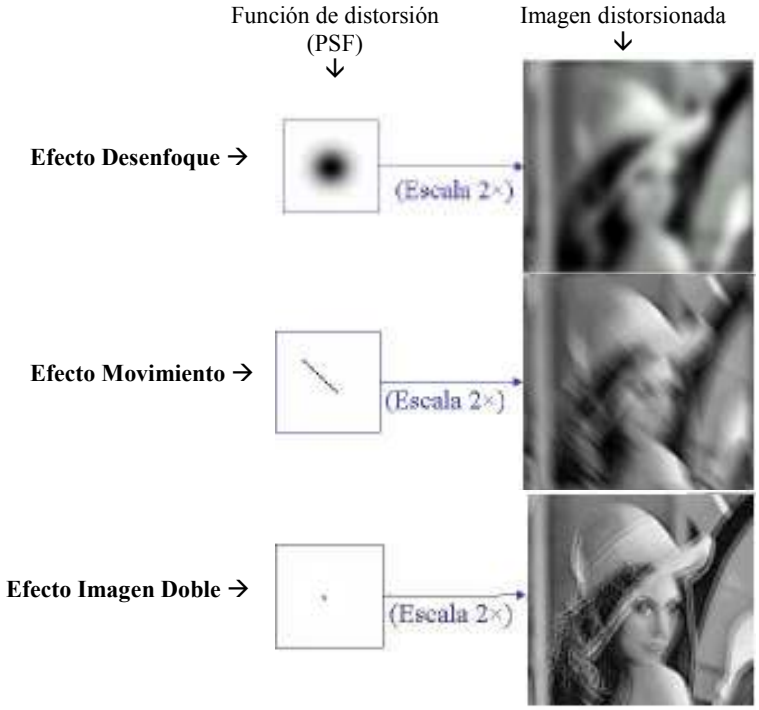


Figura 3.17: Casos típicos de distorsión (desenfoque, movimiento e imagen doble)

• Las dos técnicas más utilizadas para restaurar imágenes que sufren cualquiera de estas distorsiones son:

- *Restauración mediante filtrado inverso:*

Pasamos nuestro modelo al dominio de la frecuencia (realizando la transformada de Fourier):

$G = H \cdot F + R$	*(Las letras en mayúscula significan que estamos en el dominio frecuencial):* *G-Imagen distorsionada;* *H-Función de distorsión;* *F-Imagen sin distorsión;* *R-Ruido*

Si ahora aplicamos el filtro inverso (1/H) a los dos lados de la igualdad, nos queda:

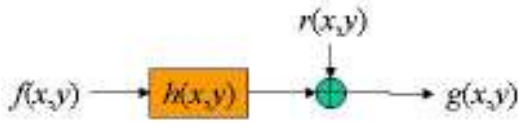
$$G/H = F + R/H$$

Es decir, si aplicamos el filtro inverso (la inversa de la función de distorsión) a la imagen distorsionada, el resultado que obtendremos será la imagen sin distorsionar más el ruido ponderado por 1/H (el filtro inverso).

Por tanto, *si la imagen tiene bastante ruido, el resultado de aplicar esta técnica no será bueno* porque estaremos obteniendo la imagen sin distorsionar más el ruido. Sin embargo, *si la imagen no tiene apenas ruido, el resultado de esta técnica será el óptimo*, puesto que estaremos recuperando la imagen sin distorsionar perfectamente (*G/H=F*).

- *Restauración mediante el filtro de Wiener:*

Esta técnica modela el proceso de degradación como un modelo estocástico:



- r: Ruido aleatorio, estacionario, independiente de la señal y de la función de autocorrelación
  $R_r(x,y) = E\{r(x_0+x, y_0+y)\, r(x_0, y_0)\}$
  y densidad espectral de potencia
  $S_r(u,v) = DFT\{R_r(x,y)\} = E\{|R(u,v)|^2\}$
- f: Imagen original (desconocida): proceso estacionario de función de autocorrelación
  $R_f(x,y) = E\{f(x_0+x, y_0+y)\, f(x_0, y_0)\}$
  y densidad espectral de potencia
  $S_f(u,v) = DFT\{R_f(x,y)\} = E\{|F(u,v)|^2\}$
- h: PSF, desconocida

Figura 3.18: Modelo estocástico del proceso de degradación de imágenes (Filtro de Wiener)

Dado este modelado, el filtro de Wiener obtiene la imagen restaurada ($f_r(x,y)$) como si de un filtro lineal (que llamaremos $w(x,y)$) se tratase.

Figura 3.19: Modelo de filtro lineal en el que se basa el Filtro de Wiener.

Los coeficientes del filtro lineal *w* (o en el dominio frecuencial *W*), se obtienen en base al siguiente criterio: se elige el filtro *W* tal que si *F=GW* es la imagen restaurada, la potencia media del error, $E\{(f_r-f)^2\}$ sea mínima.

$$W(u,v)=\frac{S_f(u,v)H^*(u,v)}{S_f(u,v)|H(u,v)|^2+S_r(u,v)}$$

Esta técnica alcanza un compromiso entre el filtrado inverso (que corrige la distorsión introducida (*h*)) y el realce del ruido (*r*). Por tanto, *para imágenes distorsionadas con presencia de ruido, ofrece mejores resultados en la restauración que el filtrado inverso* (con el filtro de Wiener controlamos el efecto del ruido en la imagen).

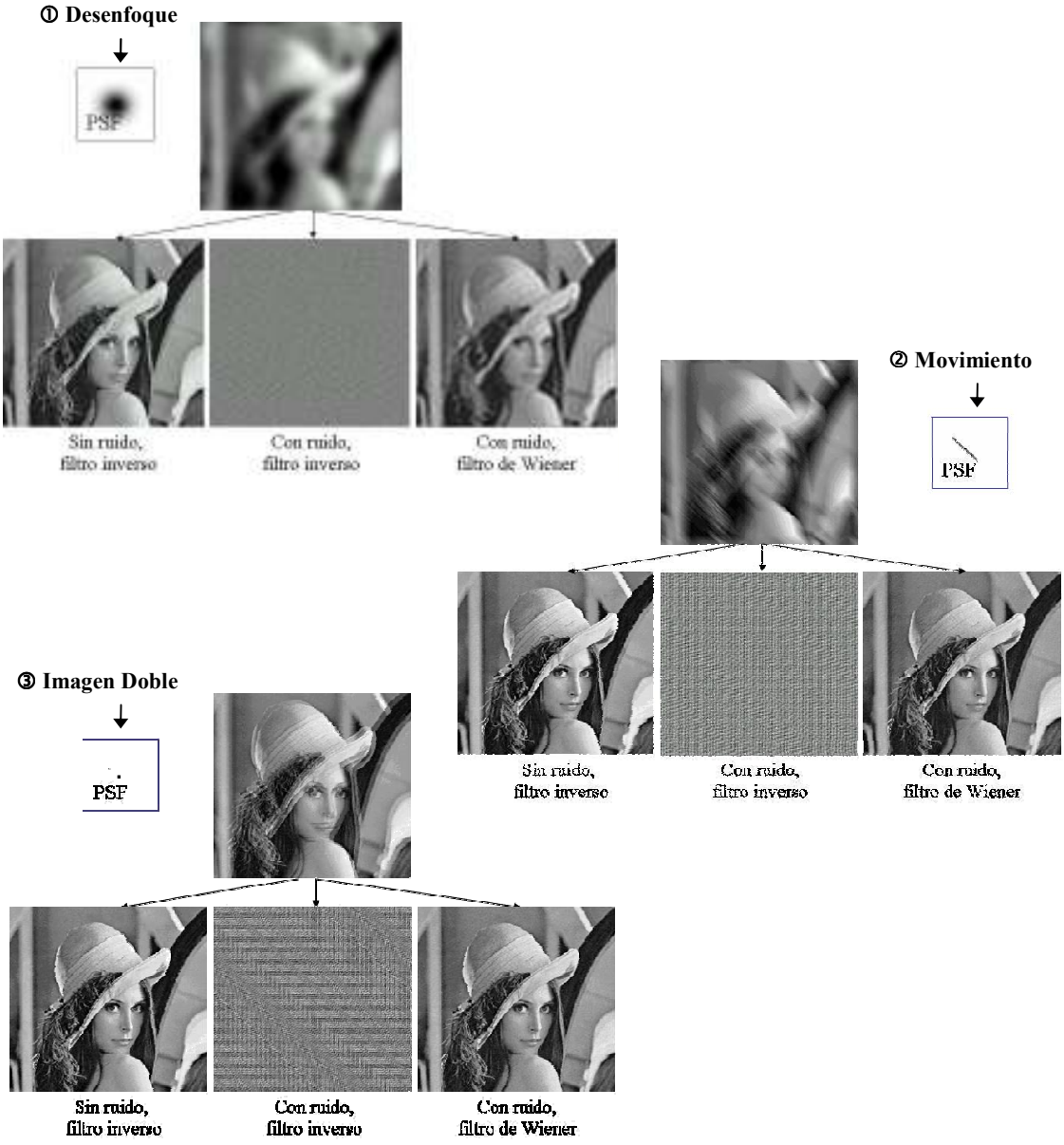


Figura 3.20: Ejemplos técnicas restauración (filtrado inverso y filtro de Wiener) para imágenes con distorsiones de tipo desenfoque ①, movimiento ② e imagen doble③.

Sin embargo, hay que puntualizar algunas consideraciones a tener en cuenta sobre el filtro de Wiener:

· No siempre se conocen S_f ni S_r: en la práctica, suele suponerse $S_r / S_f = K$ (obteniendo K a partir de información a priori o estimándola mediante exploración), de modo que la fórmula para obtener los coeficientes del filtro lineal queda:

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_r(u,v)}{S_f(u,v)}} = \frac{H^*(u,v)}{|H(u,v)|^2 + K}$$

· No siempre se conoce H

· La distorsión de la imagen no es estacionaria ni lineal

· El ruido puede depender del nivel de intensidad (no aditivo)

3.2.4. Aplicaciones de la Restauración de Imágenes

Las aplicaciones de la restauración de imágenes son numerosas. Cualquier imagen dañada (arañazos, poco contraste, ruido, pérdida de color, desenfoque…) es susceptible de ser restaurada.

El proceso de restauración sigue siendo una técnica bastante laboriosa en la mayoría de los casos. Entonces, ¿cómo delimitamos cuando nos interesa restaurar? En general, será interesante hacerlo cuando la imagen sea valiosa y la captación de la imagen no pueda repetirse, o el coste de hacerlo sea muy alto. Si hay gran cantidad de material deteriorado de interés, será viable la restauración solo si existe un procedimiento (semi)-automático para ello.

La restauración suele aplicarse en casos donde constituya una preservación del patrimonio, o cuando se hace un cambio a soporte digital de fondos de interés, o haya una demanda de transmisión en alta calidad (o de mejorar la calidad) de algún material.

En la fotografía y el cine, se utiliza para revertir el deterioro. Se plasma en la reconstrucción de fotografías (borrado de marcas de tiempo o cualquier otro defecto como los ojos rojos, etc.) o eliminando el deterioro sufrido por una película de cine antigua (manchas, parpadeos…).

En la transmisión de imágenes, por ejemplo en un streaming de video desde Internet, se utiliza para la recuperación de los bloques perdidos en la codificación y transmisión de los fotogramas. En el streaming, por fluidez del tráfico (que llegue con la suficiente rapidez ya que es prácticamente retransmisión en tiempo real), no se hacen retransmisiones de los paquetes perdidos en la transmisión (utiliza el protocolo UPD que no realiza retransmisiones por las pérdidas, en vez del protocolo TCP que si lo hace). Si las pérdidas son considerables, es muy útil tener implementado en el receptor alguna técnica de restauración (por ejemplo interpolación de fotogramas sucesivos a los correspondientes al paquete perdido que no tenemos) para disimular los efectos de las pérdidas.

4. DESCRIPCIÓN DE LA HERRAMIENTA

4. DESCRIPCIÓN DE LA HERRAMIENTA ____ 63

4.1. APLICACIÓN WEB ____ 63

4.1.2. WEB: FUNCIONAMIENTO A NIVEL DE USUARIO ____ 65

4.1.1. WEB: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES 63

4.1.3. WEB: FUNCIONAMIENTO A NIVEL TÉCNICO ____ 71

4.1.3.1. Introducción a las tecnologías implicadas en la parte web ____ 75
4.1.3.2. Funcionamiento técnico de la portada de IMAGine y resto de enlaces del índice no relacionados con cursos y autoevaluaciones ____ 79
4.1.3.4. Funcionamiento técnico de las autoevaluaciones ____ 88
4.1.3.3. Funcionamiento técnico de los cursos ____ 81
4.1.2.2. Parte 2: Cursos y autoevaluaciones ____ 69
4.1.2.1. Parte 1: Portada de IMAGine y Apartados no relacionados con cursos y autoevaluaciones. ____ 67

4.1.4. WEB: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA ____ 90

4.2. APPLETS ____ 95

4.2.2. APPLETS: FUNCIONAMIENTO A NIVEL DE USUARIO ____ 100

4.2.1. APPLETS: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES ____ 95

4.2.2.1. APPLETS DEL CURSO 'PROCESADO BÁSICO DE IMG' ____ 100

4.2.2.2. APPLETS DEL CURSO 'PROCESADO MORFOLÓGICO' ____ 109

4.2.2.3. APPLETS CURSO 'RESTAURACIÓN DE IMÁGENES' ____ 113

4.2.3. APPLETS: FUNCIONAMIENTO A NIVEL TÉCNICO ____ 119

4.2.3.1. Package BaseImagenes: Clases que se ocupan de la base de imágenes ____ 119
4.2.3.2. Package comun: Clases que contienen la funcionalidad básica de la aplicación 124
4.2.3.3. Package de los applets de los cursos ____ 135

4.2.4. APPLETS: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA ____ 140

4. DESCRIPCIÓN DE LA HERRAMIENTA

4.1. APLICACIÓN WEB

4.1.1. WEB: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES

En esta versión de IMAGine se ha realizado una reestructuración completa de toda la parte web de la aplicación.

Los principales cambios son los que mencionamos a continuación:

- La integración del contenido teórico de la asignatura de Tratamiento Digital de Imagen en formato web (XHTML) en todos los cursos (los que había ya y el nuevo de Restauración de imágenes), para así conseguir que IMAGine sea una herramienta realmente útil didácticamente hablando (el profesor podrá dar la clase utilizando la página web, ya que esta ahora integra el contenido teórico de la asignatura, además de los applets que actuarán complementando el contenido teórico con ejemplos prácticos).

- Una mejora en el diseño, más elaborado. Además, la web tendrá la misma apariencia en cualquier navegador (IE6, IE7, Firefox, etc), y también se tendrá en cuenta la resolución de pantalla del usuario para adaptar el tamaño de las fuentes (letras).

Versión 1.1 de **IMAGine**:

Versión 1.2 de **IMAGine**:

Versión 2.0 de **IMAGine** (la actual):

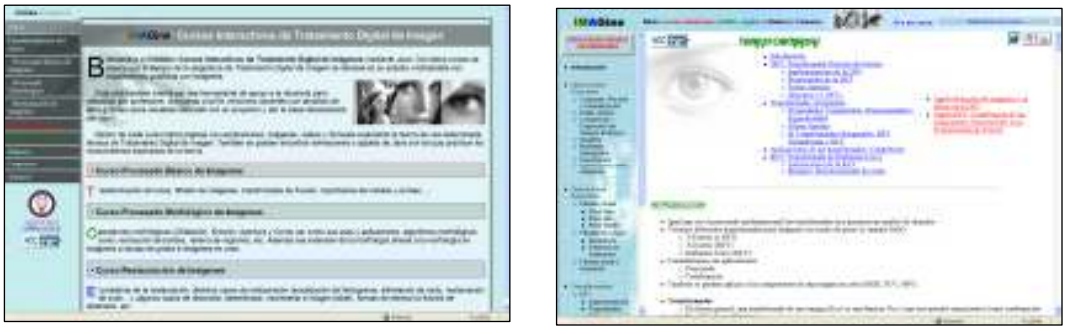


Figura 4.1: Apariencias de las versiones anteriores y la actual de IMAGine.

• Optimización de la navegación por la web, ahora más rápida y sencilla. Para ello añadimos múltiples enlaces que nos redirecionan a las páginas por las que queremos navegar y, sobre todo, en la parte de los cursos, utilizamos menús siempre visibles con todos los enlaces:

- Un menú vertical a la izquierda de la pantalla, específico del curso en el que estemos, que permitirá enlazar rápidamente cualquier parte de la teoría de dicho curso.

- Otro más genérico, situado en la parte superior de la pantalla, con enlaces a la página principal y acceso directo al resto de cursos disponibles, entre otras cosas.

- Un tercer menú, en la parte superior de la propia página que contiene la teoría, con enlaces estructurados a modo de resumen de todo el contenido teórico de esa página (en color verde), y a la derecha de estos, unos accesos directos a los applets / animaciones / contenido adicional que están contenidos en esa página.

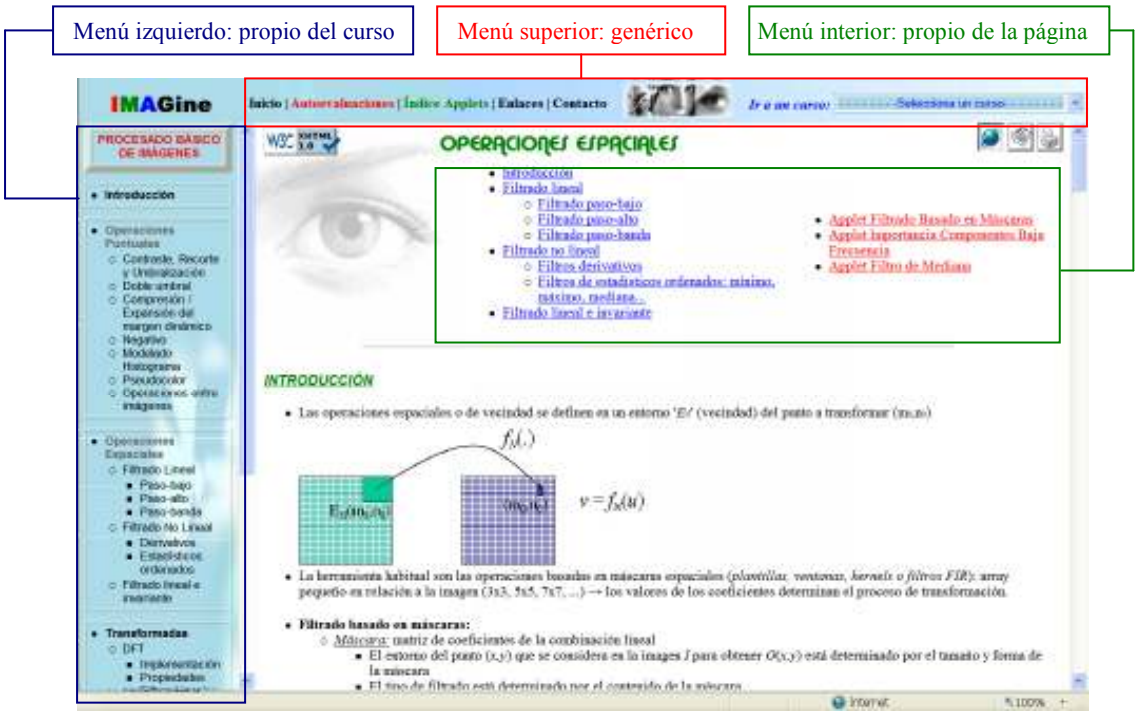


Figura 4.2: Menús de los cursos de IMAGine.

• Adaptación de IMAGine para distintos perfiles de usuarios. Distinguiremos entre una versión web, una versión docente y una vista de impresión.

Versión web: indicada para leer la página web directamente desde el ordenador (incluye explicaciones, applets y animaciones, con un tamaño de letra normal)

Versión docente: orientada para, con un proyector, visualizar la web en un aula docente, y dar la clase directamente por ahí (tiene el mismo contenido que la versión web pero con fuentes más grandes, además incluye los applets para poder hacer demostraciones en vivo durante la clase). Se omiten las explicaciones (las dará el profesor en vivo en la clase).

 *Vista de impresión:* versión para imprimir con todo el texto igual que en la versión web (sin imagen de fondo), pero sin los applets y las animaciones, aunque sí incluyendo las explicaciones.

En esta versión de IMAGine se ha hecho especialmente hincapié en el tema de las distintas versiones de la web, tratándose de un elemento realmente novedoso. Es muy interesante el hecho de que una misma web pueda tener distintas apariencias (tamaños de letra más grandes o más pequeños, algunos elementos visibles o invisibles, versión de impresión sin imágenes de fondo y sin elementos interactivos, etc.).

Lo interesante, por supuesto, es conseguir todo ello sin redundancia de información (no tiene sentido que tengamos tres páginas html con el mismo contenido donde por ejemplo solo cambie el tamaño de letra, puesto que eso complicaría enormemente el mantenimiento y actualización de la web). Por este motivo, para evitar tener información repetitiva, hemos recurrido a las hojas de estilo css y a javascript, que explicaremos en detalle más adelante.

Volviendo al punto de vista del usuario que navega por la web, las versiones serán intercambiables pulsando el icono de la versión deseada en cualquier página de los cursos de IMAGine (se muestran los 3 iconos en forma de botones en la esquina superior derecha de todas las páginas de los cursos, manteniéndose siempre visibles y apareciendo siempre pulsado el icono correspondiente a la versión seleccionada).

La versión seleccionada en una página se mantendrá seleccionada para las demás mientras no se cambie de versión (es decir, si por ejemplo estás navegando con la versión docente pulsada en el curso de procesado morfológico y cambias al curso de restauración de imágenes, seguirás navegando con la versión docente).

- Desarrollo completo del curso de Restauración de imágenes (teoría y applets).

- Autoevaluaciones de todos los cursos: para medir los conocimientos adquiridos.

4.1.2. WEB: FUNCIONAMIENTO A NIVEL DE USUARIO

La web tiene dos partes claramente diferenciadas: una con información de carácter más general o informativo y otra con contenido meramente didáctico. La primera parte es la que corresponde a la portada de IMAGine, y a la mayor parte de las páginas accesibles mediante los enlaces que en ella aparecen (‘Funcionamiento del curso’, ‘Enlaces’, ‘Autores’…). La segunda parte es la de los cursos y autoevaluaciones, con un diseño totalmente distinto más orientado a la navegación e impartición on-line de los cursos propiamente dichos (con menús de accesos rápidos a partes concretas del contenido siempre visibles, etc.).

Estas dos partes también aparecen diferenciadas por la estética, teniendo diseños distintos, pero además el tipo de contenido también es distinto. Mientras que en la primera parte aparece información general sobre IMAGine, aparte de otro tipo de información de interés, en la segunda parte aparece el contenido realmente importante de la web, es decir, los cursos on-line sobre tratamiento digital de imagen y las autoevaluaciones que servirán al alumno para medir los conocimientos adquiridos.

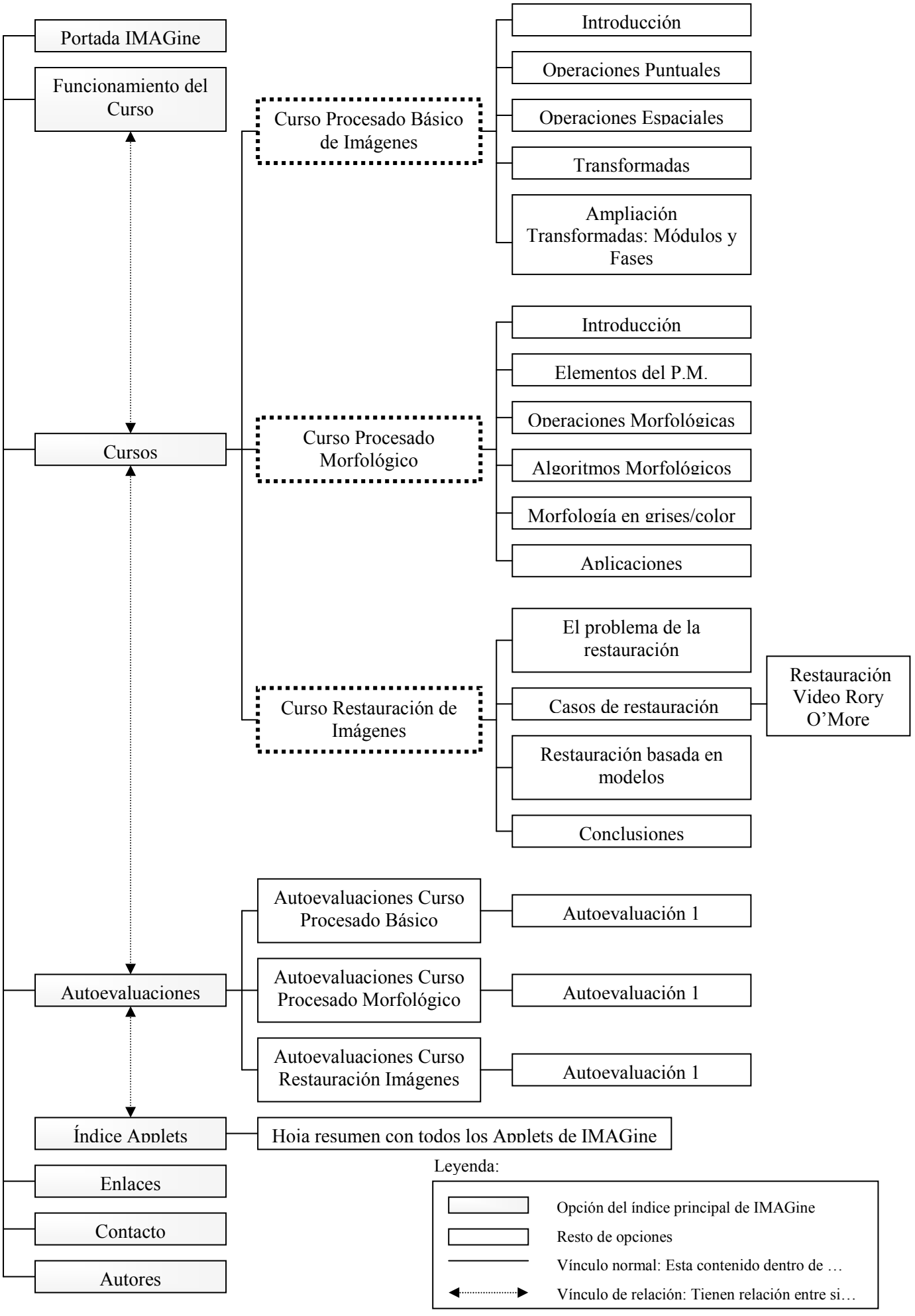


Figura 4.3: Mapa esquemático de la Web de IMAGine.

A continuación pasamos a describir en detalle cada uno de los apartados de estas partes (coincidiendo con las opciones del índice del lado izquierdo de la pantalla):

4.1.2.1. Parte 1: Portada de IMAGine y Apartados no relacionados con cursos y autoevaluaciones.

• *La portada de IMAGine ('Inicio'):* contiene un resumen del contenido general de la web, con una breve descripción de cada uno de los cursos. Además incluye un menú a la izquierda con enlaces a otras apartados con información general de IMAGine, aparte de los cursos y autoevaluaciones. Este menú se mantendrá siempre visible en el resto de páginas de la Parte 1 apareciendo sombreada en un gris más claro la pestaña en la que nos encontramos en cada momento.

Figura 4.4: 'Portada' de IMAGine.

• *'Funcionamiento del curso':* explica brevemente el funcionamiento de la Parte 2 (cursos y autoevaluaciones) de IMAGine. Se describe en qué consiste el tema de las versiones (web, docente y vista de impresión) para que el usuario pueda sacar el máximo provecho a las posibilidades que ofrece la aplicación. Explica la esquematización de la web mediante los menús (que explicábamos anteriormente). Además ofrece un applet básico para que el usuario vaya familiarizándose con este tipo de herramienta y pueda experimentar con las opciones principales que ofrece el submenú desplegable del que disponen todos los applets del curso.

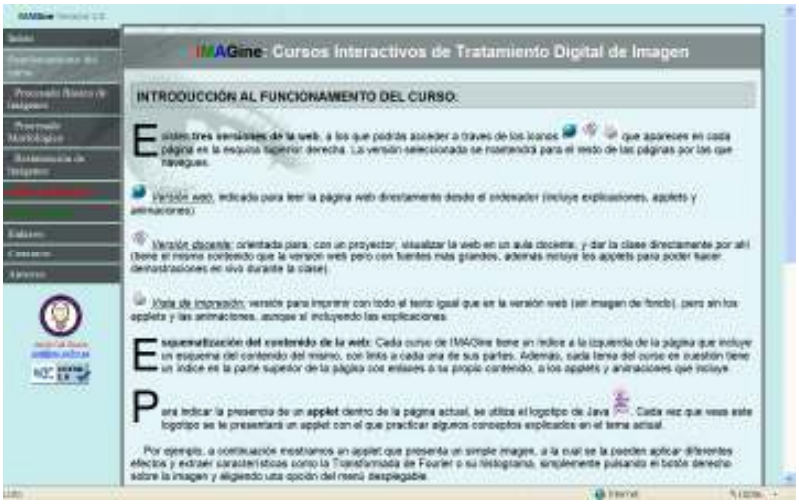


Figura 4.5 : Apartado 'Funcionamiento del curso' de IMAGine.

• *'Índice Applets':* ofrece un índice donde aparecen todos los applets que contiene la aplicación y en que parte del contenido teórico se encuentran ubicados. Para facilitar la navegación a aquellos usuarios que solo quieran utilizar los applets, al pinchar sobre cualquiera de ellos, se abre una ventana con una página donde aparecen todos los applets seguidos sin ningún tipo de contenido teórico entre medias.

Figura 4.6 : Apartado 'Índice Applets' y ventana emergente con todos los applets de IMAGine.

• *'Enlaces':* contiene una panorámica sobre el tema de tratamiento digital de imagen en Internet, con enlaces a páginas que tratan el tema en mayor o menor profundidad. También se ofrecen enlaces específicos relacionados con cada uno de los cursos de IMAGine. Algunas de las páginas se centran más en el contenido teórico y otras también tienen algún tipo de contenido interactivo (cursos, animaciones…).

Figura 4.7 : Apartado 'Enlaces' de IMAGine.

• *'Contacto':* muestra la dirección de correo del profesor-tutor que lleva este proyecto para que cualquier persona interesada en colaborar con IMAGine o que tenga algún comentario o sugerencia respecto a la web pueda transmitírselo.

• *'Autores':* contiene un resumen de las distintas versiones de IMAGine hasta la fecha y los autores de cada una de ellas, con la contribución que han aportado al proyecto.

Figura 4.8 : Apartado 'Autores' de IMAGine.

4.1.2.2. Parte 2: Cursos y autoevaluaciones

• *Cursos:* son la parte realmente interesante de la web en lo referente a contenido didáctico. Hemos separado los cursos según la temática que tratan, de tal forma que se corresponda un curso en la web con un tema de la asignatura de tratamiento digital de imagen (en algunos casos un curso en la web englobará mas de un tema de la asignatura, como en el caso del de Procesado Básico de Imágenes).

Actualmente están desarrollados los cursos de 'Procesado Básico de Imágenes' (incluyendo transformadas), 'Procesado Morfológico' y 'Restauración de imágenes'.

Todos ellos están estructurados en apartados, según un índice que se muestra al lado izquierdo de la pantalla. En el índice mencionado, aparecen agrupados los links en recuadros, donde cada recuadro corresponde a una página web distinta. Cada una de estas páginas web (que se visualizan en la parte central de la ventana, justo a la derecha del índice) contiene en la parte superior: a la izquierda un índice más detallado de la propia página (en color azul), y a la derecha unos enlaces a los applets (en rojo) y al resto de recursos contenidos en ella (en morado) si los hubiese.

Figura 4.9 : Apariencia de un 'Curso' de IMAGine.

Esta estructuración está pensada para facilitar la navegación del usuario por los cursos, para que pueda acceder rápidamente al contenido que busca (ya que en las páginas de los cursos aparecen intercaladas la teoría y los applets). Así, se podrá navegar de dos maneras por la web: siguiendo un orden secuencial (navegando de arriba abajo por los enlaces del índice del curso), o bien accediendo directamente al enlace que contiene la información de interés que estamos buscando, si lo que pretendemos es obtener una información concreta y no aprender la totalidad del curso.

Pensando en el posible uso docente de la aplicación, así como en las distintas resoluciones de pantalla de los usuarios, cada imagen y fórmula que aparece en los cursos se puede ampliar hasta su tamaño original pinchando sobre ella (aparecerá una nueva ventana con la imagen en tamaño original). Pongamos por ejemplo que un profesor está dando clase en un aula de un tamaño bastante grande, posiblemente los alumnos del fondo le soliciten que amplíe la imagen porque no ven el detalle de la foto (y esto en una web con cursos de tratamiento digital de imágenes es importante).

Por este motivo (principalmente por el uso docente de la aplicación – roles de profesores / alumnos), se ha incluido también el tema de las versiones que explicábamos en el apartado 4.1.1. de esta memoria, y que se resume principalmente en:

- Versión web: pensada para el usuario que navega por la web desde su ordenador. Contiene toda la información teórica, las explicaciones y los applets, a un tamaño de letra normal.

- Versión docente: pensada para la impartición de una clase sobre la materia por un profesor mediante un proyector donde se muestre la página web de la aplicación. Contiene toda la información teórica y los applets, pero omite las explicaciones (las dará el profesor en vivo), todo a un tamaño de letra más grande para que los alumnos lo puedan ver desde todo el aula.

- Vista de impresión: pensada para obtener un documento impreso con el contenido teórico de la web (por ejemplo, pensada para un alumno que quiera estudiarse los cursos y prefiere tener el contenido teórico en papel para poderlo subrayar, marcar, etc…). En esta versión se eliminan los componentes estéticos (imágenes de fondo…), los elementos interactivos (los applets no tiene sentido imprimirlos, son para experimentar con ellos de forma interactiva, y en papel eso no es posible, al igual que las animaciones). Se conserva la teoría, imágenes y fórmulas relacionadas con la misma, además de las explicaciones, todo ello a un tamaño de letra normal.

• *'Autoevaluaciones':* las autoevaluaciones serán la herramienta con la que el usuario pueda medir los conocimientos adquiridos en el aprendizaje de un curso. Habrá una o más autoevaluaciones por curso (inicialmente hemos incluido una), para que el usuario pueda evaluar cada curso por separado.

Todos ellos son de tipo test, pero dependiendo de la pregunta, habrá casos en las que solo una de las respuestas sea válida y otros en los que se pueda marcar más de una respuesta correcta (aumentando con ello la complejidad).

La forma de evaluar estos test es muy sencilla ya que cada pregunta está valorada con un punto. Dependiendo del tipo de pregunta (si es multirespuesta o no) se puntuara de forma distinta según explicamos a continuación:

- Si la pregunta es del tipo en las que solo se puede elegir una respuesta como correcta, si la aciertas, sumas un punto, y si la fallas sumarás cero puntos.

- Si la pregunta es multirespuesta (de las que puede haber una o más respuestas correctas), sumarás tantas fracciones del punto que vale toda la pregunta como respuestas correctas hallas marcado (por ejemplo, si hay dos respuestas correctas, y tú solo has marcado una de ellas, sumarás 0.5 puntos)

Finalmente, al enviar la autoevaluación, aparecerá un mensaje donde se mostrarán dos notas. La primera de ellas es la correspondiente a sumar todos los puntos obtenidos en todas las preguntas (teniendo en cuenta que cada pregunta vale 1 punto), y la segunda es la nota obtenida sobre 10 (independientemente del número de preguntas que contenga el formulario).

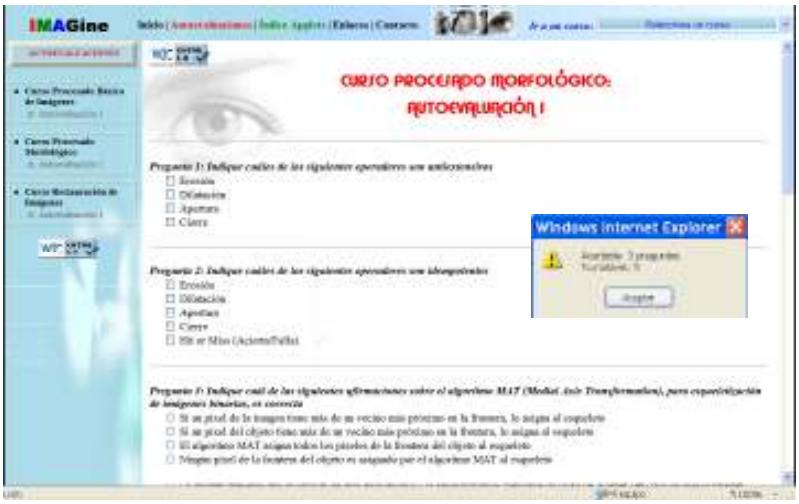


Figura 4.10 : Apariencia de una 'Autoevaluación' de IMAGine.

4.1.3. WEB: FUNCIONAMIENTO A NIVEL TÉCNICO

A la hora de crear la web, hemos tratado de emplear los estándares existentes en la actualidad. El uso de hojas de estilo css combinado con código html de acuerdo al estándar XHTML, nos ayuda a separar el contenido de la apariencia (facilitando así la accesibilidad y la automatización). Javascript nos sirve para implementar pequeñas funcionalidades más complejas que nos permiten interactuar más con el usuario y personalizar la aplicación.

- El uso de estándares facilita la accesibilidad de una web, entendiendo por accesibilidad la posibilidad de que un producto o servicio web pueda ser accedido y usado por el mayor número posible de personas, indiferentemente de las limitaciones propias del individuo o de las derivadas del contexto de uso.

Según el W3C (consorcio internacional encargado de producir estándares para la World Wide Web): "la accesibilidad de una web queda asegurada cuando cualquier usuario desde cualquier tipo de navegador y sin tener en cuenta sus conocimientos y habilidades, es capaz de obtener una total comprensión de la información presentada así como una interacción plena con el sitio web".

Hay que tener en cuenta que para muchas personas Internet es una herramienta que puede facilitar tareas que de otra forma podrían ser más costosas o arduas de conseguir o de llevar a cabo, por lo que una página Web accesible a un público más amplio (por ejemplo accesible también a personas con discapacidad) facilita esa tarea a un número más elevado de gente, que de esa forma se puede beneficiar también de esas herramientas.

Otra de las ventajas de diseñar la web según los estándares, es que ofrece más garantías de que ésta se visualice correctamente en cualquier dispositivo o navegador, o en cualquier caso, los ajustes para conseguirlo serían mucho menores.

Para saber si una web cumple cualquiera de los estándares, podemos recurrir a los validadores de la W3C: *http://validator.w3.org/*. Si una página web pasa la validación, podrá ser marcada con el símbolo identificativo que le proporciona el validador:

Figura 4.11: Etiquetas de validaciones sobre estándares referentes a páginas web que ofrece la W3C.

* *Para ampliar información sobre accesibilidad web, se puede consultar en el* ***'Capítulo 10: Bibliografía'*** *de esta memoria el 'Apartado 4.1-Accesibilidad Web'.*

- Para conseguir que IMAGine tuviese la misma apariencia independientemente del navegador utilizado, aparte de diseñar una página en base a los estándares (cosa que sin duda ha facilitado en gran medida la tarea), hemos tenido que recurrir a ciertas 'trucos'.

Primero de todo, hay que mencionar que Internet Explorer, sobre todo en las versiones anteriores a la 7, hace una interpretación del código (por ejemplo con algunos atributos de las hojas de estilo css) que no se ajusta a lo definido en los estándares, lo que hace necesario ajustar las páginas web para visualizarlas correctamente en navegadores IE6 o inferior, respecto al resto de navegadores como Firefox, Ópera, etc., que sí interpretan correctamente los estándares.

Pongamos un ejemplo muy conocido de este defecto de versiones anteriores de Internet Explorer 7: el modelo de caja del CSS describe cómo ciertos elementos de una página web son mostrados por un navegador gráfico, es decir, permite a los elementos contenidos dentro de la caja ser envueltos por un relleno (*padding*), *bordes* y *márgenes*. Para la CSS publicada por W3C, cuando se define un ancho de un elemento dentro de la caja, este ancho debería determinar sólo el ancho del contenido dentro de la caja, con el relleno, bordes y márgenes agregados después. IE versión 5 incorrectamente incluye el relleno y bordes dentro de la anchura especificada, resultando en un modelo mucho más angosto cuando es visualizada.

Otro fallo que nos ha resultado especialmente molesto, es que IE6 e inferiores no reconocen el atributo definido por css que deja un elemento, por ejemplo una imagen, en una posición fija de la pantalla, aunque se mueva la barra de desplazamiento: *position:fixed*. Esto nos ha obligado ha implementar una función en javascript que detecte el navegador y la versión utilizada, y si es IE6 o inferior, realice una especie de 'truco' para conseguir un efecto similar.

Veamos, por poner un ejemplo, la función que ha sido necesaria implementar para corregir este defecto de los navegadores IE6 o inferior:

```
//Hacemos esto para solucionar que versiones anteriores de Internet Explorer 7 no reconocen position:fixed
if (navigator.appName == "Microsoft Internet Explorer")
{
	var version = navigator.appVersion.substring(navigator.appVersion.indexOf('MSIE') + 5,
			navigator.appVersion.indexOf('MSIE') + 6);
	if (version<7)
	{
		//Reescribimos el estilo de div.versiones para que los iconos de cambiar las versiones aparezcan
		//en una posicion fija en la pantalla
		document.write ("<style>div.versiones{position: absolute;top: expression( 5 + (ignorar =
		document.documentElement.scrollTop ? document.documentElement.scrollTop :
		document.body.scrollTop) + \'px\' );right:0px;width:105px;}</style>");
	}
}
```

• Hemos tenido en cuenta también la resolución de pantalla establecida por el usuario en su ordenador a la hora de modificar los tamaños de las letras (para una resolución de 800x600, un tamaño de letra *large* puede que se vea demasiado grande, pero para una resolución de 1400x900 es el tamaño de letra ideal para un título o para los links más importantes del menú).

La función de JavaScript que nos ha servido para modificar los tamaños de letra según la resolución de pantalla que tenga el usuario y el navegador empleado es la siguiente:

```
/**Modificamos el tamaño de las fuentes según el navegador y la resolución de pantalla del usuario */
//Navegador Internet Explorer
if (navigator.appName == "Microsoft Internet Explorer")
{
	document.write ("<style>div#versionIMAGine{font-size:small}
				div#links a{font-size:medium}
				div#uc3m{font-size:x-small}
			</style>");
	//Si la resolución de pantalla horizontal es igual o inferior a 1030
	if (screen.width<=1030)
	{
		document.write ("<style>div#versionIMAGine{font-size:xx-small}
					div#links a{font-size:small}
					div#uc3m{font-size:xx-small}
				</style>");
	}
 }
//Resto de Navegadores (no Internet Explorer)
else
{
	document.write ("<style>div#versionIMAGine{font-size:small}
				div#links a{font-size:large}
				div#uc3m{font-size:x-small; top:480px;}
				div#validacionW3C{top:587px;}
			</style>");
	//Si la resolución de pantalla horizontal es igual o inferior a 1030
	 if (screen.width<=1030)
	 {
		document.write ("<style>div#versionIMAGine{font-size:small}
					div#links a {font-size:large}
					div#uc3m{font-size:x-small}
				</style>");
	}
}
```

• Utilizaremos enlaces relativos en todo IMAGine, es decir, nunca pondremos la dirección completa (enlaces absolutos), sino que pondremos una ruta que parta del directorio donde nos encontremos. El navegador es el encargado de construir la ruta absoluta que corresponda.

Con los enlaces relativos conseguimos que no haga falta modificar nada del código de la aplicación para usarlo/probarlo en local (ubicado en un directorio cualquiera del ordenador), ni para subirlo al servidor de Internet y probarlo en la web (se podrá subir sin realizar ninguna modificación).

Ejemplos:

- Enlace relativo (en el que se suben dos directorios respecto al que estamos poniendo ../ por cada directorio que se quiera subir):

```
<a href="../../imagenes/operacionesPuntuales1.jpg" >
```

- Enlace relativo (en el que se quiere ir a directorios contenido en el que estamos).

```
<a href="/Contenido/Introduccion/Introduccion.html" target="contenido">
<a href="./Contenido/Introduccion/Introduccion.html" target="contenido">
```

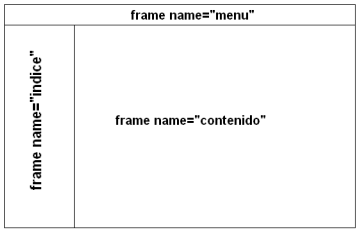
- Enlace absoluto (con la dirección completa).

```
<a href="www.tsc.uc3m.es/imagine/index.html">
```

• La maquetación de la web es distinta para la portada y enlaces no relacionados con autoevaluaciones y cursos que para estos cursos y autoevaluaciones.

- *Maquetación de la portada y enlaces no relacionados con autoevaluaciones*: no hay frames que dividan la pantalla, por lo que cada página corresponde a una página html, pero donde cada una de ellas tiene elementos similares al resto. Para ello dividiremos las páginas en recuadros con un identificador (por ej), que luego situaremos en la posición de la pantalla que queramos mediante una hoja de estilo común a todas estas páginas (ej declaración en pagina css: div#linksIndice {position:fixed ;top: 35px; left:0px ; width: 170px;}). Habrá algunos de estos recuadros que contengan la misma información en todas las páginas, por ejemplo el recuadro que contiene el índice o el de la versión de IMAGine (ya que estos elementos tendrán que estar siempre visibles en todas las páginas). Otros, tendrán la información propia de cada página ( ), y aunque se deberán llamar igual en todas las páginas, en cada uno el contenido será distinto.

- *Maquetación de los cursos y autoevaluaciones:* habrá un index.html por cada curso / página de autoevaluaciones que divida la pantalla en 3 frames y defina las proporciones de cada uno.



El frame de la izquierda se llamará *indice* y contendrá el índice del curso o de las autoevaluaciones; el frame superior se llamará *menú* y contendrá el menú genérico de la aplicación; y por último, el frame central se llamará *contenido* y contendrá las páginas web con la información del curso propiamente dicho.

Para indicar en un elemento <a href> que queremos que el contenido del enlace se muestre en uno u otro frame habrá que indicarlo en el atributo target.

4.1.3.1. Introducción a las tecnologías implicadas en la parte web

Haremos una breve descripción de los principales lenguajes de programación empleados en la creación de la parte web.

- ***XHTML:***

XHTML es un lenguaje de etiquetado hipertextual extensible. Parte de HTML 4.0 (el lenguaje utilizado para la elaboración de páginas web) redefiniéndolo en base a XML.

En la actualidad, se definen tres versiones, más o menos estrictas, de XHTML:

- XHTML 1.0
  - XHTML 1.0 Strict: no se permite el uso de capacidades de presentación de html, para ello se debe emplear CSS.
  - XHTML 1.0 Transitional: permite algunas etiquetas de formato dentro del código de la página para ser soportada por navegadores antiguos. Se puede incluir CSS para hacer pequeños ajustes.
  - XHTML 1.0 Frameset: Permite el uso de marcos (frames) para dividir la ventana del navegador.
- XHTML 1.1: nueva versión que parte de XHTML 1.0 strict e incluye nuevos requisitos, eliminando totalmente elementos relacionados con el estílo. Se presenta dividido en módulos.
- XHTML Basic: versión simplificada de XHTML 1.1 para dispositivos limitados.
- XHTML 2.0: añade nuevos módulos al ya modularizado XHTML 1.1 para así intentar minimizar el uso de scripts dentro de los documentos XHTML.

Para validar si una página web está diseñada según cualquiera de estos estándares, podemos recurrir a los validadores del w3c (*http://validator.w3.org/*).

Esta versión de IMAGine es válida por *XHTML 1.0 Transitional*, lo que quiere decir que cumple las reglas básicas de XHTML, como son por ejemplo, una correcta estructura de etiquetas (bien anidadas, siempre con etiqueta de cierre…), nombres de elementos y atributos en minúsculas, valores de los atributos entrecomillados, caracteres válidos (por ejemplo la *ñ* se escribe *ñ* y las tildes, por ejemplo *é*, como *é*), etc.

En previsión a futuras mejoras de IMAGine, la mayoría del código está orientada a ser validada con versiones más estrictas que XHTML 1.0 Transitional, ya que apenas hemos utilizando etiquetas de presentación (apenas algún atributo *border* o elemento *center*), empleando en su lugar hojas de estilo css.

- ***Hojas de estilo CSS:***

Las hojas de estilo en cascada (Cascading Style Sheets) constituyen un mecanismo simple para controlar el estilo y formato de los documentos html y xml, separando el contenido de la presentación. Describen cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o como se va a visualizar en cierto tipo de dispositivo o en otro.

Además, son muy importantes para unificar la apariencia de toda la web, y son necesarias para una buena navegación: es imprescindible que elementos similares tengan la misma apariencia, el mismo estilo (los índices y subíndices siempre con un determinado estilo y color, los applets con recuadros iguales que permitan localizarlos rápidamente…).

Pero no solo eso, con las hojas de estilo también podemos definir distintas maneras de visualizar nuestra web, es lo que llamaremos versiones (en nuestro caso hemos definido tres versiones de acuerdo a nuestras preferencias: web/docente/impresión). Digamos que puedes definir una versión identificándola con una hoja de estilo, en la que cierto elemento (por ejemplo los títulos que tengan la etiqueta h2) se vean de determinada manera (por ejemplo en color verde), y otra versión identificándola con otra hoja de estilo, en la que ese mismo elemento se vea de otra forma, o incluso que no se vea.

CSS funciona a base de reglas, declaraciones sobre el estilo de uno o más elementos del documento. Las reglas se construyen siguiendo la estructura:

elemento { propiedad: valor; propiedad: valor;}

Ej. *h2 { color: blue;}* → todas las etiquetas h2 que estén vinculadas a esta hoja de estilo se verán afectadas por esta declaración, y por tanto se mostrarán en azul.

Hay tres formas de incluir código css en un documento (aplicándole un estilo):

① Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento <link>, y que se encuentra ubicada en la ruta indicada en el atributo ref. El elemento link debe ir situado en la sección <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.O Transitional//EN"
"http://www.w3.org/TR/xhtml11//DTD/xhtml11-transitional.dtd">
<html>
<head>
        <title>Hoja de estilo externa</title>
        <link rel="stylesheet" type="text/css" href="../../../HojasEstilo/EstiloVersionWeb.css"/>
</head>
<body>
  .
</body>
</html>
```

② Utilizando el elemento <style>, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección <head>. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.O Transitional//EN"
"http://www.w3.org/TR/xhtml11//DTD/xhtml11-transitional.dtd">
<html>
<head>
        <title>Hoja de estilo interna</title>
        <style type="text/css">
        body {
          font-family: Georgia, "Times New Roman", serif;
          color:red;
          background-color: rgb(0,100,100);
        }
        h1 {
          color: blue;
        }
        </style>
</head>
<body>
        <h1>Aqui va el t&iacute;tulo en azul</h1>
</body>
</html>
```

③ Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo <style> dentro de <body>. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

```
<h1 style="color:blue;">Aqui va el t&iacute;tulo en azul</h1>
```

En IMAGine hemos utilizado la forma número ① a la hora de incluir las hojas de estilo, principalmente porque así, si hay que efectuar cualquier modificación, solo será necesario hacerla en un sitio (en la hoja css) y se aplicará a todas las páginas vinculadas a ella (lo que facilita enormemente el desarrollo y mantenimiento de la web).

Además, de esta forma, podemos cambiar dinámicamente la hoja de estilo que se aplica a una página web en un momento dado, ya que solo es necesario cambiar la línea que vincula la hoja de estilo a aplicar (así se implementa el diseño de las versiones web/docente/impresión).

** Para más información sobre HTML, XHTML y CSS, se puede consultar en el '**Capítulo 10: Bibliografía'** de esta memoria el 'Apartado 4.1-HTML, XHTML y CSS'.*

- ***Javascript:***

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web, normalmente construidas con lenguaje HTML.

Se utiliza para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Por tanto, Javascript es una herramienta muy útil a la hora de implementar ciertas funcionalidades más complejas, por ejemplo implementar una función que cambie de hoja de estilo al pulsar sobre un botón de la web, o mostrar mensajes de tipo alert con información al producirse algún evento (por ejemplo, un alert que muestre la nota obtenida al completar uno de los formularios de autoevaluación). También existen funciones que detectan el navegador utilizado por el usuario o la configuración de pantalla, y en base a ello se pueden cambiar parámetros de la apariencia de la web para que visualmente esta se vea igual que en el ordenador donde fue diseñada.

De manera similar a lo que ocurre con las páginas css, existen varias formas de incluir código JavaScript en las páginas web:

① Utilizando una archivo externo con extensión .js que estará vinculado a la página web a través del elemento <script>, ubicado en la ruta indicada dentro del atributo src. El elemento script debe ir situado en la sección <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.O Transitional//EN"
"http://www.w3.org/TR/xhtml11//DTD/xhtml11-transitional.dtd">
<html>
<head>
        <title>JavaScript externo</title>
        <script language="JavaScript" type="text/javascript" src="../../../Scripts/ScriptContenido.js">
        </script>
</head>
<body>…
</body>
</html>
```

② Insertando el código JavaScript incrustrado dentro de la propia página html, dentro de los elementos de html <script></script>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.O Transitional//EN"
"http://www.w3.org/TR/xhtml11//DTD/xhtml11-transitional.dtd">
<html>
<head>
        <title>JavaScript interno</title>
        <script language="JavaScript" type="text/javascript">
        if (navigator.appName=="Microsoft Internet Explorer"){
                alert( "Estas usando Internet Explorer");
                document.write ("<style>body{width:105px;}</style>");
        }
        </script>
</head>
<body>
  .
</body>
</html>
```

Para IMAGine, al igual que ocurría con las hojas de estilo css, también hemos preferido utilizar la opción ①, que es la enlaza un archivo externo con el código javascript. De esta forma, cualquier código nuevo que haya que insertar o cualquier modificación del existente, solo habrá que hacerlo sobre el archivo externo, y automáticamente se reflejará en todas las páginas web que tengan vinculado ese archivo.

Respecto a la sintaxis del lenguaje JavaScript, comentaremos los rasgos y elementos principales:

- Es un código semejante a Java y C, pero no orientado a objetos, sino más bien basado en prototipos (sin herencia, donde las nuevas clases se crean clonando clases prototipo y ampliando su funcionalidad).

- Elementos→variables: son espacios en memoria asociados a un valor, un objeto o a una función. Hay varios tipos de variables: string, number, object, function, array, boolean ... En JavaScript se pueden definir variables explícitamente, si van precedidas por la palabra *var* [Ej. var numEj=0;], o implícitamente si no van precedidas por ella [Ej. numEj=0;].

- Elementos→operadores: existen diversos tipos de operadores, comunes en la mayoría de los lenguajes de programación

  º Operador de asignación [=]: Ej. var numEj = 5;

  º Operadores aritméticos [+ ; - ; * ; / ; % ; ++ ; --]: Ej. var numEj = 5 + 10;

  º Operadores relacionales [< ; > ; >= ; <= ; == ; ¡=]: Ej. if (numEj >= 5)

  º Operadores lógicos [&& ; || ; ! ; ^ ; etc.]: Ej. if (numEj>= 5 && numEj<=10)

  º Operadores especiales [?: ; this ; instanceOf ; etc.] : Ej. var newNum= numEj!=10? numEj : 0 ;

-Elementos→objetos del lenguaje: JavaScript permite crear nuestros propios objetos, pero cuenta con una serie de ellos predefinidos con sus propios métodos ya creados [*String*; *Array*; *Boolean*; *Date*; *Function*; *Number*; *Object*; *Option*; ...].

-Elementos→objetos del navegador: JavaScript está orientado principalmente a páginas web, por lo que cuenta con objetos predefinidos con sus métodos correspondientes que hacen referencia al navegador y algunas de sus características/propiedades [*Jerarquia*; *Window*; *Frame*; *Location*; *History*; *Navigator*; *Document*; *Link*; *Anchor*; *Image*].

-Elementos→objetos relacionados con formularios: uno de los usos más extendidos de JavaScript es la de manipular y validar formularios. Por ello tiene una serie de objetos predefinidos, con sus respectivos métodos, que facilitan esta tarea [*Form*, *Text*, *Password*, *Button*, *Checkbox*, *Radio*; *Select*; *Hidden*; etc.]

-Estructuras de control: formadas por los bucles y sentencias condicionales más comunes en la mayoría de los lenguajes de programación [*if...else...*; *while*; *do...while*; *for*; *for..in*; *switch*]

Veamos un ejemplo sencillo de tratamiento de formularios con JavaScript:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.O Transitional//EN"
"http://www.w3.org/TR/xhtml11//DTD/xhtml11-transitional.dtd">
<html>
<head>
        <title>Ejemplo de JavaScript</title>
        <script LANGUAGE="JavaScript">
                <!--
                function Mostrar()
                {
                    alert('Su nombre: ' + formulario.nombre.value);
                    alert('El password: ' + formulario.pass.value);
                }
                //-->
        </script>
</head>
<body>
        <form action="" name="formulario" id="formulario" method="get">
        Nombre: <input type="text" name="nombre" value="Tu nombre" maxlength="15"/><br/>
        Password: <input type="password" name="pass" maxlength="10"/><br/>
        </form>
        <a href="javascript:Mostrar();">Mostrar datos</a><br/>
</body>
</html>
```

** Para más información sobre JavaScript, se puede consultar en el '**Capítulo 10: Bibliografía'** de esta memoria el 'Apartado 4.1-JavaScript'.*

4.1.3.2. Funcionamiento técnico de la portada de IMAGine y resto de enlaces del índice no relacionados con cursos y autoevaluaciones

Dentro de este apartado se incluyen las páginas contenidas en el directorio /Portada/ y el index.html que cuelga del directorio padre.

Como indicábamos en la parte de *maquetación*, estas páginas no se muestran en pantallas divididas en frames, por lo que hay parte de código redundante entre ellas para mantener la apariencia de que se está navegando por un menú (siempre visible) donde sólo cambia el contenido del panel de la derecha. Se ha diseñado así para dar un efecto

visualmente más llamativo que si hubiésemos empleado frames. De esta forma hemos conseguido el efecto de transparencia de la imagen de fondo respecto a las capas superiores, dando la impresión de que el recuadro que contiene la información se desliza sobre el fondo de la página, manteniéndose el índice estático mientras esto sucede. Todos estos efectos se han conseguido con las hojas de estilo, combinando distintos tipos de posiciones para las imágenes de fondo de cada elemento de tipo div de la página.

No obstante, hemos empleado otros elementos de maquetación que han facilitado enormemente la tarea: usamos elementos div con un identificador (los elementos div son como recuadros que agrupan una serie de elementos) y que luego podemos situar en la pantalla directamente desde una hoja css común a todas estas páginas. Todas las páginas de este apartado tendrán prácticamente idénticos todos los elementos div, exceptuando el que tiene el identificador , que será el que contenga la información que se mostrará en el panel deslizante de la derecha.

• La hoja de estilo externa que está enlazada a todas estas páginas es /HojasEstilo/EstiloPortada.css. En ella viene indicado en que parte de la pantalla se sitúa cada elemento div, que fuentes e imágenes de fondo utilizaremos en cada uno, etc.

• La hoja de JavaScript externa que está enlazada a todas estas páginas es /Scripts/ScriptPortada.js. Servirá para cambiar automáticamente el tamaño de las fuentes según el navegador y la resolución de pantalla con la que se esté visualizando la web. También hemos metido el código para que funcione el Google Analytics [http://www.google.com/analytics/es-ES/] y podamos ver estadísticas de utilidad sobre IMAGine (accesos, tipo de conexión, navegador utilizado…).

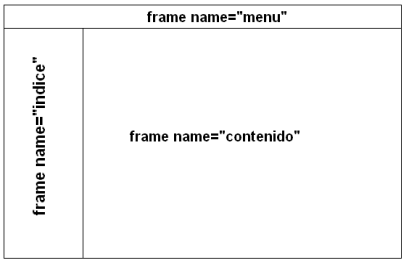
ScriptPortada.js:

```
<!--
/** Modificamos el tamaño de las fuentes según el navegador y la resolución de pantalla del usuario */

//Navegador Internet Explorer
if (navigator.appName == "Microsoft Internet Explorer")
{
        document.write ("<style>div#versionIMAGine{font-size:small}div#links a{font-size:medium}
                                        div#uc3m{font-size:x-small}</style>");

        //Si la resolución de pantalla horizontal es igual o inferior a 1030
        if (screen.width<=1030)
        {
            document.write ("<style>div#versionIMAGine{font-size:xx-small} div#links a{font-size:small}
                                        div#uc3m{font-size:xx-small}</style>");
        }
}
//Resto de Navegadores (no Internet Explorer)
else
{
        document.write ("<style>div#versionIMAGine{font-size:small} div#links a{font-size:large}
                                        div#uc3m{font-size:x-small; top:480px;} div#validacionW3C{top:587px;} </style>");

       //Si la resolución de pantalla horizontal es igual o inferior a 1030
       if (screen.width<=1030)
       {
            document.write ("<style>div#versionIMAGine{font-size:small} div#links a {font-size:large}
                                        div#uc3m{font-size:x-small} </style>");
       }
}

/** Esto es para que funcione el Google Analyst */
_uacct = "UA-2823541-1";
urchinTracker();

-->
```

4.1.3.3. Funcionamiento técnico de los cursos

Dentro de este apartado se incluyen las páginas contenidas en el directorio /Curso_xxxxxx/ y en /Menu/Menu.html.

Recordemos que los cursos están divididos en tres frames según muestra la imagen de la derecha.



El funcionamiento y contenido de cada frame es distinto, por lo que pasaremos a explicarlos por separado.

❶ Funcionamiento técnico del frame *menu*:

La página que se muestra en el frame *menú* está en /Menu/Menu.html, y es común a todos los cursos (siempre se mostrará el mismo menú horizontal en todos los cursos).

Básicamente está compuesta de enlaces '*a href*' que nos llevan a las páginas de la portada de IMAGine (Inicio, Autores, etc.), a las autoevaluaciones o a otros curso de nuestra web.

Estos últimos, los enlaces a otros cursos de IMAGine, se encuentran en un combo, donde al seleccionar cualquiera de las opciones (cada uno de los cursos es una opción) te redirige a la página principal del curso pulsado. Así, cuando se incluyan nuevos cursos en IMAGine, tan solo habrá que añadir mas opciones al combo, y no será necesario reestructurar el menú para que, por ejemplo, cupiesen todos los nombres de los cursos horizontalmente en el menú.

Para implementar la funcionalidad del combo con los enlaces a los cursos, ha sido necesario un pequeño fragmento de JavaScript, ya que dentro de las opciones de un elemento *select* no se pueden incluir enlaces. Este código consiste en incluir dentro del elemento *select* el evento *onchange*, que se se invocará cada vez que se pulse una opción del *select*. Hemos puesto que cuando se llame a dicho evento, la acción a realizar será invocar el link cuyo valor viene definido en el atributo *value* del elemento *option* que se ha seleccionado (de entre los disponibles en el combo). La nueva página se mostrará en toda la pantalla (deshaciendo la estructura de frames).

```
Ir a un curso:
<select onchange="parent.location.href=this.value">
        <option value=""> - - - - - - - -Selecciona un curso- - - - - - - -</option>
        <option value="../Curso_ProcesadoBasico/index.html">Curso Procesado B&aacute;sico de
Im&aacute;genes</option>
        <option value="../Curso_ProcesadoMorfologico/index.html">Curso Procesado Morfologico</option>
        <option value="../Curso_RestauracionImagenes/index.html">Curso Restauraci&oacute;n de
Im&aacute;genes</option>
</select>
```

• La hoja de estilo externa que está enlazada a la página que se muestra en el frame *'menu'* es /HojasEstilo/EstiloMenu.css , que se encarga de definir la presentación de los elementos de esta página (color y tipo de fuente, imagen de fondo, estilo de los links según si se han visitado ya o no, etc).

• La hoja de JavaScript externa que está enlazada a esta página es /Scripts/ScriptMenu.js. En ella simplemente se cambia el tamaño de las fuentes según el navegador y la resolución de pantalla con la que se esté visualizando la web.

ScriptMenu.js:

```
<!--
/** Modificamos el tamaño de las fuentes según el navegador y la resolución de pantalla del usuario */

//Navegador Internet Explorer
if (navigator.appName == "Microsoft Internet Explorer"){
	document.write ("<style>table{font-size:medium} h2{font-size:x-large}</style>");

	//Si la resolución de pantalla horizontal es igual o inferior a 1030
	if (screen.width<=1030){
		document.write ("<style>table{font-size:xx-small} h2{font-size:medium}</style>");
	}
}
//Resto de Navegadores (no Internet Explorer)
else{
	document.write ("<style>table{font-size:large} h2{font-size:x-large}</style>");

	//Si la resolución de pantalla horizontal es igual o inferior a 1030
	if (screen.width<=1030){
		document.write ("<style>table{font-size:small} h2{font-size:medium}</style>");
	}
}
-->
```

❷ Funcionamiento técnico del frame *indice*:

La página que se muestra en el frame *indice* está en /Curso_xxxx/Indice/Indice.html.

Contiene un esquema del curso en cuestión con enlaces directos a cada una de sus partes. Aparecerán dentro de un recuadro todos los enlaces que se encuentren dentro de la misma página html (normalmente el curso se divide en varios apartados, correspondiendo cada uno de ellos a una página html).

Las páginas a las que hacen referencia todos estos enlaces se mostrarán en el frame *contenido*, por lo que habrá que añadir a los elementos '*a href*' el atributo '*target="contenido"*'. Recordemos que la idea es que el índice se muestre siempre en el lado izquierdo de la pantalla (frame *indice*) y en la parte central (frame *contenido*) se muestre el contenido propiamente dicho del curso.

Por ejemplo, mostramos un apartado del índice de un curso:

```
<table border="1">
<tr><td>
   <ul>
   <li><a href="../Contenido/AmpliacionTransformadas/AmpliacionTransformadas.html" target="contenido">
        <b>Ampliaci&oacute;n Transformadas: M&oacute;dulos y Fases</b></a>
            <ul>
            <li><a href="../Contenido/AmpliacionTransformadas/AmpliacionTransformadas.html#fase"
                  target="contenido"> Importancia Fase</a></li>
            <li><a  href="../Contenido/AmpliacionTransformadas/AmpliacionTransformadas.html#oppenheim"
                  target="contenido">Experimento Oppenheim</a></li>
            <li><a href="../Contenido/AmpliacionTransformadas/AmpliacionTransformadas.html#promedio"
                   target="contenido">Promedio m&oacute;dulos y fases</a></li>
            </ul>
   </li>
   </ul>
</td></tr>
</table>
```

• La hoja de estilo externa que está enlazada a la página que se muestra en el frame *'indice'* es /HojasEstilo/EstiloIndice.css. En ella se define todo lo relativo a la presentación (imagen de fondo, fuentes para el título con el nombre del curso y para los apartados del mismo, estilo para los links cuando se pasa por encima de ellos/cuando se han visitado ya/cuando todavía no han sido pulsados ninguna vez …).

• La hoja de JavaScript externa que está enlazada a todas las páginas que se muestran dentro del frame '*contenido*' es /Scripts/ScriptIndice.js. Se encarga de ajustar los márgenes de los línks del índice respecto a los bordes de la pantalla dependiendo del navegador con el que se esté visualizando la web. Además modifica el tamaño de las letras según el navegador y resolución empleada.

ScriptIndice.js:

```
<!--
/**
 * Para actualizar el contenido de la web con las paginas en la versión
 * seleccionada (web, docente o de impresión)
 * (Actualmente no se usa, pero se deja implementado por si en un futuro hiciese falta)
 */
function cargarWebVersiones(paginaInicio,version){
        parent.contenido.location.href="../Contenido/" + paginaInicio + "/" + paginaInicio + "_" + version + ".html";
        parent.indice.location.href="Indice_"+version+".html";
        if (version=="VistaImpresion"){
                alert ("Pinche ahora sobre los links del índice para acceder a la vista de impresión");
        }
}

/** Adaptamos la presentación de la página según el navegador y la resolución de pantalla del usuario */
//Navegador Internet Explorer
if (navigator.appName == "Microsoft Internet Explorer"){
        //Ponemos los margenes a las listas del indice
        document.write ("<style>div#indice ul {margin:2px 2px 2px 20px;}</style>");

        //Si tenemos una resolucion baja ponemos las fuentes mas peques
        if (screen.width<=1030 && screen.height<=780){
                document.write ("<style>div#indice table{font-size:xx-small;}
                                        div#indice h4 {font-size:xx-small;}</style>");
        }
}
//Resto de navegadores
else{
        //Ponemos los margenes a las listas del indice
        document.write ("<style>div#indice ul {margin:2px 2px 2px -25px;}</style>");

        //Si tenemos una resolucion alta ponemos las fuentes mas grandes
        if (screen.width>1030 && screen.height>780){
                document.write ("<style>div#indice table{font-size:medium;}
                                        div#indice h4 {font-size:large;}</style>");
        }
}
-->
```

❸ Funcionamiento técnico del frame *contenido*:

Las páginas que se muestran en el frame *contenido* están en el directorio /Curso_xxxx/Contenido/.

Estas páginas son las que contienen la teoría del curso. Se caracterizan por tener en la parte superior un índice con links internos al contenido de la propia página html. En la parte superior, en el lado derecho, también existe otro índice con enlaces al contenido interactivo (applets, animaciones…) que se encuentran en esa página.

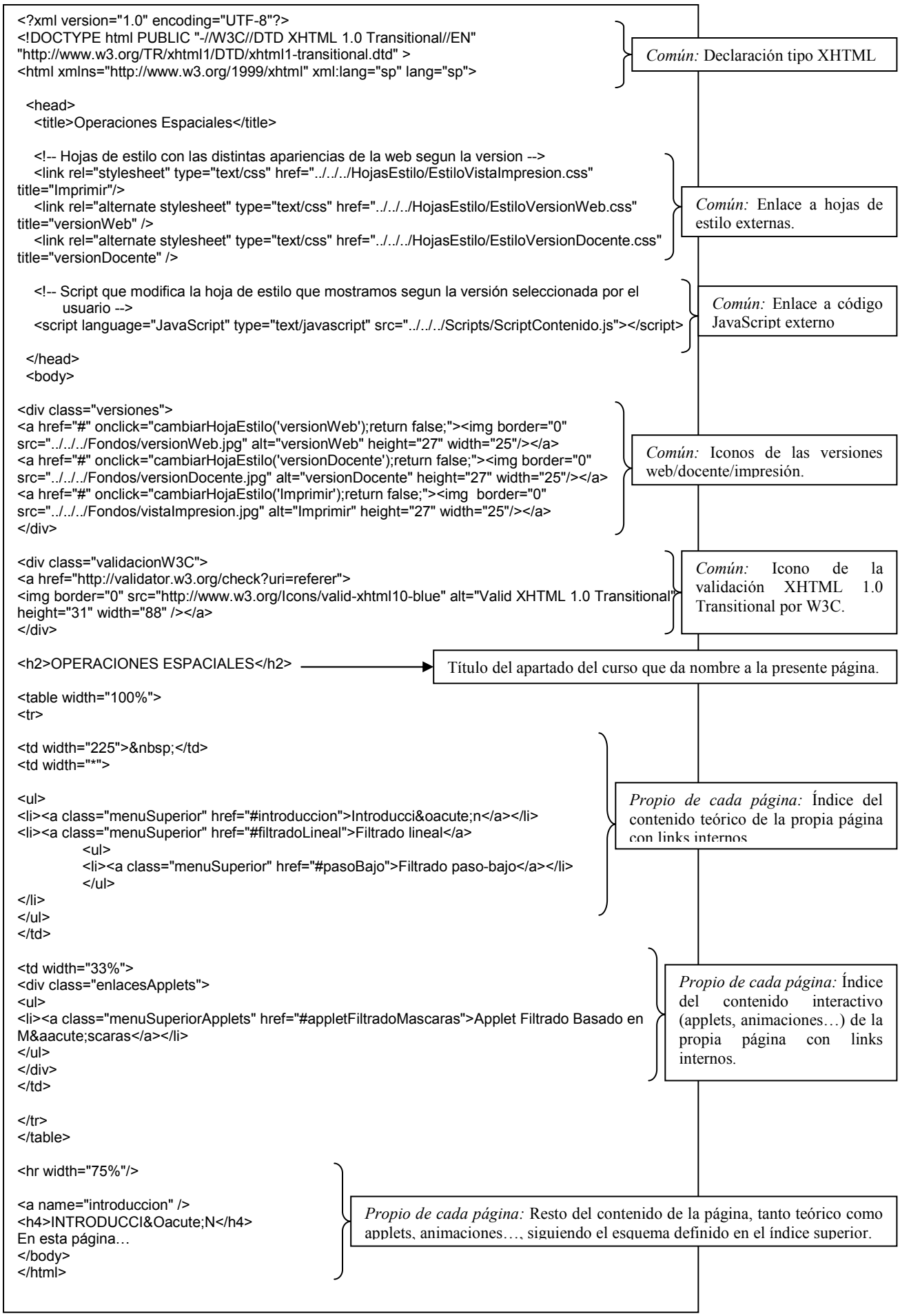


Figura 4.12: Esquema de las secciones de las páginas XHTML que contienen la teoría de los cursos.

Aparte de estos índices en la parte superior, todas ellas contienen elementos idénticos: los enlaces a las hojas de estilo y de JavaScript externas, el icono de validación de la W3C como página XHTML 1.0 Transitional, el recuadro con los 3 iconos de las versiones para poderlas cambiar por el usuario, etc. El contenido teórico, los applets y las animaciones son las cosas que realmente variarán de unas páginas a otras.

Es necesario intentar utilizar los elementos definidos para cada objeto en las hojas de estilo, unificando así la presentación (Ej: incluir los applets entre <div class= "applets>).

• Las hojas de estilo externas que están enlazadas a todas las páginas que se muestran en el frame *'contenido'* son /HojasEstilo/EstiloVersionWeb.css; /HojasEstilo/EstiloVersionDocente.css; /HojasEstilo/EstiloVistaImpresion.css. Todas ellas contienen las declaraciones de los mismos elementos, lo que varía son los atributos y valores de los mismos dentro de ellos. Así, podemos definir que el elemento div.applets en la versionWeb tenga una apariencia, en la versionDocente otra y que en la vistaImpresion directamente desaparezca, y así con el resto de elementos. De esta forma conseguimos implementar el mencionado efecto de las 3 versiones mencionadas.

• La hoja de JavaScript externa que está enlazada a todas las páginas que se muestran dentro del frame '*contenido*' es /Scripts/ScriptContenido.js.

Crea la función que permite solucionar el problema con el estilo *position:fixed* de las hojas de estilo para versiones anteriores a Internet Explorer 7. Así se verán los iconos de las versiones siempre en la misma posición en la pantalla.

Además contiene todas las funciones necesarias para poder implementar el tema de las versiones web/docente/impresión. La funcionalidad que deben tener consiste principalmente en:

- Cambiar la hoja de estilo que está utilizando la web en ese momento al pulsar sobre uno de los botones de las versiones, estableciendo el del botón pulsado.

- Dibujar un recuadro sobre la imagen del botón de la versión que está seleccionada en cada momento, como si se tratase de un botón pulsado, y sobre el resto de versiones no seleccionadas, dibujar recuadros como si fuesen botones sin pulsar.

- Implementar la carga/descarga de cookies a la hora de cargar la página o salir de ella. Cuando se carga la página, se obtiene el cookie con la versión seleccionada (web/docente/impresión) que había en la última página que visitamos, y se establece a la página actual. Cuando salimos de una página, se guarda en el cookie qué versión teníamos seleccionada en ese momento, para que al cargar posteriormente una nueva página, sigamos navegando con dicha versión.

ScriptContenido.js:

```
// Hacemos esto para solucionar que versiones anteriores de Internet Explorer 7 no reconocen
// position:fixed
if (navigator.appName == "Microsoft Internet Explorer"){
        var version = navigator.appVersion.substring(navigator.appVersion.indexOf('MSIE') + 5,
                                                     navigator.appVersion.indexOf('MSIE') + 6);
        if (version<7){//Reescribimos el estilo de div.versiones para que los iconos de cambiar las
                       //versiones aparezcan en una posicion fija en la pantalla
                document.write ("<style>div.versiones{position: absolute;top: expression( 5 + (ignorar =
document.documentElement.scrollTop ? document.documentElement.scrollTop : document.body.scrollTop) + \'px\'
);right:0px;width:105px;}</style>");
        }
}
```

```
//---------- IMPLEMENTACION VERSIONES WEB/DOCENTE/IMPRESION ----------

//Variables que definen los nombres de las versiones
var versionWeb="versionWeb";
var versionDocente="versionDocente";
var vistaImpresion="Imprimir";

//Variable que almacena la última versión seleccionada distina de vistaImpresion
var ultimaVersionSelect="";

/**
 * Función para cambiar la hoja de estilo segun la opcion seleccionada por el usuario
 * @param title-> valor del atributo title de elemento <link rel> de la página html
 * donde está linkada la hoja de estilo que queremos establecer.
 */
function cambiarHojaEstilo(title) {
	var i, j, a, b;
	//Recorremos todos los elementos link de la pagina web
	for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
		//Comprobamos los elementos link que en su atributo rel tienen la palabra style
		if(a.getAttribute("rel").indexOf("style") != -1) {
			//Deshabilitamos el resto de hojas de estilo
			a.disabled = true;
			//Habilitamos la hoja de estilo cuyo title es el que hemos pasado por parámetros
			if(a.getAttribute("title") == title) {
				a.disabled = false;
				if (title!=vistaImpresion){
					pintarVersionSeleccionada(title);
					ultimaVersionSelect = title;
				}
			}
		}
	}
	//Si la versión seleccionada es imprimir, imprimiremos la página con esa hoja de estilos pero
	//luego la cambiaremos a la hoja de estilos que habia seleccionada previamente
	if (title==vistaImpresion){
		window.print("#");
		if (navigator.appName == "Microsoft Internet Explorer"){
			if (ultimaVersionSelect!=""){
				cambiarHojaEstilo (ultimaVersionSelect);
			}else{
				cambiarHojaEstilo (versionWeb);
			}
		}else{
			pintarVersionSeleccionada(vistaImpresion);
		}
	}
}

/**
 * Pinta recuadros al estilo de botones sobre las imagenes que indican las versiones.
 * Se pinta como boton pulsado el que contiene la version seleccionada
 */
function pintarVersionSeleccionada(versionSelect){
	for(j=0; ( b= document.getElementsByTagName("img")[j]); j++) {
		if (b.getAttribute("alt")==versionSelect){
			b.style.border='0.4mm inset rgb(240,240,240)';
		}else if (b.getAttribute("alt")!=versionSelect && (b.getAttribute("alt")==versionWeb
			||b.getAttribute("alt")==versionDocente||b.getAttribute("alt")==vistaImpresion))
		{
			b.style.border='0.4mm outset rgb(240,240,240)';
		}
	}
}

/**  Devuelve la hoja de estilo en uso */
function getActiveStyleSheet() {
	var i, a;
	for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
		if(a.getAttribute("rel").indexOf("style") != -1&& a.getAttribute("title")&& !a.disabled)
			return a.getAttribute("title");
	}
	return null;
}
```

```
/**
 * Crea un cookie
 * @ param name -> nombre del atributo que queremos guardar en el cookie
 * @ param value -> valor del atributo cuyo nombre hemos pasado en name
 * @ param days -> dias de validez del cookie
 */
function createCookie(name,value,days) {
	if (days) {
		var date = new Date();
		date.setTime(date.getTime()+(days*24*60*60*1000));
		var expires = "; expires="+date.toGMTString();
	}
	else expires = "";
	document.cookie = name+"="+value+expires+"; path=/";
}

/**
 * Lee de un cookie el valor del atributo cuyo nombre le pasamos
 * por parámetros
 * @param name -> nombre del atributo que queremos leer del cookie
 * @return value -> valor del atributo cuyo nombre hemos pasado por parámetros
 */
function readCookie(name) {
	var nameEQ = name + "=";
	var ca = document.cookie.split(';');
	for(var i=0;i < ca.length;i++) {
		var c = ca[i];
		while (c.charAt(0)==' ') c = c.substring(1,c.length);
		if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
	}
	return null;
}

function getPreferredStyleSheet() {
	var i, a;
	for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
		if(a.getAttribute("rel").indexOf("style") != -1
		&& a.getAttribute("rel").indexOf("alt") == -1
		&& a.getAttribute("title"))
			return a.getAttribute("title");
	}
	return null;
}

/**
 * Funcion que se llama al cambiar de pagina y que guarda el nombre de la hoja de estilo
 * que habia seleeccionada como un atributo de un cookie.
 */
window.onunload = function(e) {
	var versionSelect = getActiveStyleSheet();
	createCookie("style", versionSelect, 365);
}

/**
 * Funcion que se llama al cargar la página y carga la hoja de estilo
 * que habia seleccionada anteriormente leyendo el atributo correspondiente del cookie
 */
window.onload = function(e) {
	//Leemos la hoja de estilo guardada en el cookie con el atributo style.
	 var cookie = readCookie("style");
	//Si hemos obtenido algun valor al leer el cookie, se lo asignamos a la variable versionSelectAnterior,
	//pero si no, ponemos un valor por defecto.
	var versionSelectAnterior = cookie ? cookie : getPreferredStyleSheet();

	//Ponemos a la página actual la hoja de estilo que había seleccionada en la última página visitada del curso
	if (versionSelectAnterior!=vistaImpresion && (versionSelectAnterior==versionWeb  ||
	    versionSelectAnterior==versionDocente))
	{
		cambiarHojaEstilo (versionSelectAnterior);
	}
	//Si no había ninguna hoja de estilo seleccionada anteriormente, ponemos por defecto la
	//correspondiente a la versionWeb
	else{
		cambiarHojaEstilo (versionWeb);
	}
}
-->
```

4.1.3.4. Funcionamiento técnico de las autoevaluaciones

Las páginas relativas a las autoevaluaciones están en el directorio /Autoevaluaciones/.

Estas páginas siguen la misma maquetación que los cursos, por lo que estarán divididas en tres frames (*índice*, *menú* y *contenido*). Los frames *índice* y *menú* funcionan de manera idéntica a los homónimos en las páginas de los cursos, por lo que no pasaremos a explicarlos de nuevo.

En cuanto a las páginas que se muestran en el frame *contenido*, son las que se encuentran en el directorio /Autoevaluaciones/Contenido/, y en ellas se encuentran formularios con preguntas y respuestas relativas a la teoría dada en los cursos de IMAGine, que harán las veces de autoevaluaciones.

En estos formularios, podemos incluir preguntas con posibles respuestas para cada una de estas preguntas. El usuario se encargará de seleccionar la(s) que considere correcta(s).

Las posibles respuestas irán incluidas en elementos html de tipo *radio* o *select* → en estos casos sólo habrá una respuesta correcta a la pregunta, que marcaremos con el atributo *value="bien"*; el resto de respuestas erróneas llevarán el atributo *value= "mal"*.

También se pueden incluir preguntas multirespuesta, donde hay una o varias posibles respuestas correctas → en estos casos utilizaremos elementos de html de tipo *checkbox*, marcando con el atributo *value="bien"* las respuestas correctas, y además incluyendo en todas las posibles respuestas de una misma pregunta el atributo *name* con el mismo valor en todas ellas (por ejemplo *name= "pregunta1"*).

Por último, habrá que modificar en la etiqueta html de tipo <form> incluida en la página, el valor del segundo parámetro de la funcion averiguarNota() poniendo en él el número de preguntas que tiene el formulario (esto nos servirá para calcular la nota sobre 10 obtenida).

• La hoja de estilo externa que está enlazada a todas las páginas que se muestran dentro del frame '*contenido*' es /HojasEstilo/EstiloVersionWeb.css, ya que la estética de la página es similar a la versión web de los cursos, y aquí no es necesario tener la versión docente ni la de impresión. El resto de hojas de estilo que se enlazan a las páginas que se muestran en los otros frames son las mismas que para el caso de los cursos.

• La hoja de JavaScript externa que está enlazada a todas las páginas que se muestran dentro del frame '*contenido*' es /Scripts/ScriptFormulario.js. Servirá para calcular la nota obtenida tras rellenar el formulario de autoevaluación. Partiendo de la base de que cada pregunta vale un punto, se encarga de contar el número de respuestas correctas de la siguiente manera:

- En las preguntas con respuestas de tipo *radio* o *select* (monorespuestas): cuenta las respuestas marcadas por el usuario cuyo atributo *value* tiene el valor *"bien"* (Sumando por cada una que cumpla estas condiciones un punto).

- En las preguntas con respuestas de tipo *checkbox* (multirespuestas): primero cuenta el número de respuestas correctas que hay (atributo *value="bien"*) y divide el punto que vale esa pregunta entre el número de respuestas correctas, obteniendo una

fracción del punto que vale esa pregunta. Por cada respuesta correcta marcada se sumará dicha fracción, de tal forma que si se seleccionan todas las respuestas correctas se sumará un punto.

Por último, obtendremos dos notas. La primera de ellas será el resultado de la suma de todos los puntos obtenidos, y la segunda será la nota sobre diez, obtenida tras dividir tu puntuación entre el número de preguntas que tiene el formulario y multiplicarlo por diez. Ambas se mostrarán al usuario en un mensaje emergente al enviar la autoevaluación.

ScriptFormulario.js:

```
/**
 * Funcion para averiguar la nota obtenida en una autoevaluacion.
 * Una vez calculada la muestra en un alert (mensaje emergente)
 * @param formulario -> formulario de la web con las preguntas y respuestas marcadas
 * @param numPreguntas -> numero de preguntas que tiene la autoevaluacion
 */
function averiguarNota(formulario, numPreguntas){
	//Preguntas acertadas
	var aciertos=0;
	//Nota sobre 10 sacada en el cuestionario
	var nota;
	//Variables para contabilizar las preguntas de tipo checkbox
	var numOpcionesBienEnPregunta;
	var numAciertosEnPregunta;
	for (i=0;i<formulario.length;i++){
		//Obtenemos los aciertos en las preguntas de tipo botones radio (Solo una opcion valida)
		if (formulario.elements[i].type=="radio"){
			If (formulario.elements[i].value=="bien" && formulario.elements[i].checked){
				aciertos++;
			}
		}
		//Obtenemos los aciertos en las preguntas de tipo botones checkbox (Una o mas de una
		//opcion valida)
		if (formulario.elements[i].type=="checkbox"){
			//Entramos aqui cuando estamos en el primer elemento de una nueva pregunta de
			//tipo checkbox
			if ((i==0) || (formulario.elements[i].name!=formulario.elements[i-1].name)){
				//Inicializamos las variables
				numOpcionesBienEnPregunta=0;
				numAciertosEnPregunta=0;
			}
			if ( formulario.elements[i].value=="bien"){
				numOpcionesBienEnPregunta++;
				if ( formulario.elements[i].checked){
					numAciertosEnPregunta++;
				}
			}
			//Entramos aqui cuando estamos en el ultimo elemento checkbox de una pregunta
			if (i<formulario.length &&
(formulario.elements[i].name!=formulario.elements[i+1].name)){
				//Sumamos a aciertos la parte acertada de la pregunta
				aciertos+=numAciertosEnPregunta/numOpcionesBienEnPregunta;
			}
		}
		//Obtenemos los aciertos en las preguntas de tipo select-one
		if (formulario.elements[i].type=="select-one") {
			if (formulario.elements[i].value=="bien"){ aciertos++; }
		}
		//Obtenemos los aciertos en las preguntas de tipo texto
		if (formulario.elements[i].type=="text"){
			if (formulario.elements[i].value=="a"){ aciertos++; }
		}

	}
	nota=aciertos/numPreguntas*10;
	alert ("Acertaste: " +		aciertos + " preguntas.\nTu nota es: " + nota);
}
```

4.1.4. WEB: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA

Lo ideal es conservar la estética de la web para futuras ampliaciones. Para ello se deberá ojear las posibilidades que incluyen las **hojas de estilo**.

La dinámica que hemos empleado, es crear una hoja de estilo para cada parte de IMAgine, así habrá una hoja de estilo para la portada, otra para el menú, otra para el índice, y por último, tres para los cursos (versionWeb, versionDocente, vistaImpresion).

Todas ellas tienen elementos comunes, por ejemplo, en la mayoría se definen unos estilos que afectan a las fuentes, para evitar así utilizar etiquetas de estilo de fuentes en las páginas web. De esta forma, cuando se quiera que IMAGine pase una validación más estricta que la de XHTML 1.0 Transitional, será mucho más fácil (Por ejemplo, lo correcto sería poner <h2 class="cursiva">Esto va en cursiva</h2> en vez de <h2>Esto también va en cursiva</h2>)

/*CLASES QUE AFECTAN A LAS FUENTES*/

```
/* Clase para los subindices de las formulas */
.fuentePeque {font-size:xx-small;}

/* Clase para la negrita */
.negrita {font-weight:bold;}

/* Clase para la cursiva */
.cursiva {font-style: oblique;}

/* Clase para la italica */
.italica {font-style: italic; font-weight:bold;}

/* Clase para la subrayada */
.subrayada {text-decoration: underline;}

/* Clase para la cursiva subrayada */
.cursivaSub {font-style: oblique; text-decoration: underline;}

/* Clase para la negrita subrayada */
.negritaSub {font-weight:bold; text-decoration: underline;}

/* Clase para la italica subrayada */
.italicaSub {font-style: italic; font-weight:bold; text-decoration: underline;}
```

Además, hay que tener en cuenta que cuando se incluya un nuevo elemento en la hoja de estilo versionWeb.css para alguna cosa contenida en las páginas de los cursos, habrá que incluirla también en la versión docente y en la vista de impresión (para mantener la dinámica de las 3 versiones), pero indicando si para esa versión en concreto queremos que haga algo distinto (por ejemplo que en la vista de impresión no se vea, etc.…). Si no queremos que haga nada distinto, con copiar el nuevo elemento tal cual en las 3 versiones bastaría.

Como consejo, sugerimos que a la hora de crear una nueva página, se copie una existente que tenga una estética similar a la que nosotros deseamos, y vayamos modificando lo que haga falta. Es decir, si por ejemplo queremos incluir un applet, lo ideal es ir a una página donde se incluya uno y ver como lo hace, copiando el código, y modificándolo para que muestre el applet y el texto que nosotros queremos.

4.1.4.1. Crear una nueva página en la parte de la portada

Para crear una nueva página en la parte de portada o enlaces no relacionados con cursos/autoevaluaciones, habrá que:

- Crear una página .html dentro del directorio /portada/ como copia de alguna de las existentes en ese directorio.

- Modificar el contenido que hay dentro de la etiqueta con el contenido que desees que tenga tu propia página.

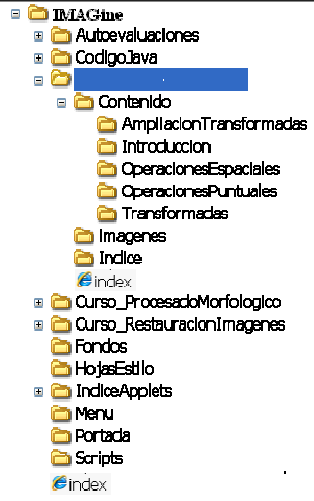
- Modificar en la página index.html y en las páginas que se incluyen dentro del directorio /portada/ el elemento que contiene el menú, para incluir en él el enlace a la nueva página. Los enlaces de este menú están dentro del recuadro en todas estas páginas. Tener cuidado de poner el estilo sombreado al enlace de la nueva página cuando incluyas el menú en tu nueva página (y que el resto se queden con estilo no sombreado), ya que debe aparecer sombreado el enlace en el que te encuentres en cada momento.

```
<div id="links">
   <a class="menu" href="index.html">Inicio</a>
   <a class="menu" href="./Portada/Funcionamiento.html">Funcionamiento del curso</a>
   <a class="cursos" href="./Curso_ProcesadoBasico/index.html"><span style="color:red;">&#8226;</span>
Procesado B&aacute;sico de Im&aacute;genes</a>
   <a class="cursos" href="./Curso_ProcesadoMorfologico/index.html"><span
style="color:green;">&#8226;</span> Procesado Morfol&oacute;gico</a>
   <a class="cursos" href="./Curso_RestauracionImagenes/index.html"><span style="color:blue;">&#8226;</span>
Restauraci&oacute;n de Im&aacute;genes</a>
   <a class="menu" href="./Autoevaluaciones/index.html"><span style="color:red;">Autoevaluaciones</span></a>
   <a class="menu" href="./IndiceApplets/IndiceApplets.html"><span style="color:green;">&Iacute;ndice
Applets</span></a>
   <a class="menu" href="./Portada/Enlaces.html">Enlaces</a>
   <a class="menu" href="./Portada/Contacto.html">Contacto</a>
   <a class="menu" href="./Portada/Autores.html">Autores</a>
   <a class="menuSelect" href="./Portada/NewLink.html">Nuevo link</a>
</div>
```

4.1.4.2. Crear un curso nuevo

Para crear un curso nuevo habrá que:

- Crear un directorio con el nombre del nuevo curso en el directorio padre (donde están las carpetas de los otros cursos, por ejemplo el de /Curso_ProcesadoBasico/). Recomendamos copiar el directorio de un curso de los ya existentes para mantener la estructura, e ir renombrando lo archivos que nos hagan falta dentro de los directorios finales.



Todos los cursos contienen un index.html donde viene definida la estructura de frames y las páginas que aparecerán por defecto en ellos al pinchar en el enlace del curso.

También tienen el directorio /imágenes/ con todas las imágenes del curso, y el directorio /Indice/, con el índice propio del curso.

Por último está el directorio /Contenido/, con tantos subdirectorios como apartados tenga el curso, correspondiendo cada uno a una página web. Dentro de cada uno irá la página html con la información de ese apartado.

Figura 4.13: Estructura de directorios de un curso de IMAGine.

- Modificar en la página index.html (del directorio padre) y en las páginas que se incluyen dentro del directorio /portada/ el elemento que contiene el menú, para incluir en él el enlace al nuevo curso. Los enlaces de este menú están dentro del recuadro en todas estas páginas.

```
<div id="links">
   <a class="menuSelect" href="index.html">Inicio</a>
   <a class="menu" href="./Portada/Funcionamiento.html">Funcionamiento del curso</a>
   <a class="cursos" href="./Curso_ProcesadoBasico/index.html"><span style="color:red;">&#8226;</span>
Procesado B&aacute;sico de Im&aacute;genes</a>
   <a class="cursos" href="./Curso_ProcesadoMorfologico/index.html"><span
style="color:green;">&#8226;</span> Procesado Morfol&oacute;gico</a>
   <a class="cursos" href="./Curso_RestauracionImagenes/index.html"><span style="color:blue;">&#8226;</span>
Restauraci&oacute;n de Im&aacute;genes</a>
   <a class="cursos" href="./Curso_NuevoCurso/index.html"><span style="color:red;">&#8226;</span>
Nuevo curso </a>
   <a class="menu" href="./Autoevaluaciones/index.html"><span style="color:red;">Autoevaluaciones</span></a>
   <a class="menu" href="./IndiceApplets/IndiceApplets.html"><span style="color:green;">&Iacute;ndice
Applets</span></a>
   <a class="menu" href="./Portada/Enlaces.html">Enlaces</a>
   <a class="menu" href="./Portada/Contacto.html">Contacto</a>
   <a class="menu" href="./Portada/Autores.html">Autores</a>
```

- Modificar en la página /Menu/Menú.html el <select></select> donde se puede elegir el curso al que se quiere acceder, añadiendo la opción del nuevo curso indicando el directorio donde se encuentra.

```
Ir a un curso:
<select onchange="parent.location.href=this.value">
        <option value="">- - - - - - - -Selecciona un curso- - - - - - - -</option>
        <option value="../Curso_ProcesadoBasico/index.html">Curso Procesado B&aacute;sico de
Im&aacute;genes</option>
        <option value="../Curso_ProcesadoMorfologico/index.html">Curso Procesado Morfologico</option>
        <option value="../Curso_RestauracionImagenes/index.html">Curso Restauraci&oacute;n de
Im&aacute;genes</option>
        <option value="../Curso_NuevoCurso/index.html">Curso Nuevo Curso</option>
</select>
```

- Crear los que es el curso propiamente dicho: borrando las imágenes del directorio /Curso_NuevoCurso/imágenes/ y añadiendo las propias del curso; modificando el /Curso_NuevoCurso/Indice/indice.html con la estructura del índice del nuevo curso; crear las páginas de los subapartados del curso siguiendo el estilo de las de otros subapartados de otros cursos (a la hora de poner los títulos y subtítulos, de insertar un applet, un ejercicio o código de Matlab…), dentro de Curso_NuevoCurso/Contenido/*Nombre_subapartado/Nombre_subapartado.html*.

- Se deberán incluir los applets insertados en el nuevo curso en la página con el índice de applets (ubicada en /IndiceApplets/IndiceApplets.html) y en la página donde están todos los applets juntos (/IndiceApplets/Contenido/Applets.html) de la manera en la que están incluidos los demás (poniendo los nombres de los applets creados en el lugar correspondiente).

- No olvidar añadir los .class de las clases java creadas o modificadas en el directorio /CodigoJava/

- Por último, se deberá crear una autoevaluación del curso nuevo, de la manera que se describe en el punto "Para crear una autoevaluación nueva de un curso nuevo" de este manual rápido.

4.1.4.3. Crear una autoevaluación nueva dentro de un curso existente

Para crear una autoevaluación nueva dentro de un curso existente:

- Crear una autoevaluacion.html como copia de otra que haya en el directorio de autoevaluaciones, en el subdirectorio del curso de la que sea la autoevaluación (Directorio: /Autoevaluaciones/Contenido/*NombreCurso/Autoevaluacion1.html*).

- Para modificar la autoevaluación.html con nuevas preguntas, hay una serie de normas que hay que cumplir para que se pueda calcular la nota obtenida con el script que hay para tal fin:

· Modificar las preguntas por las que deseemos poner. Todas las opciones que ofrecemos como posibles respuestas (puede ser cualquier número de respuestas) para una misma pregunta deberán incluir el atributo *name* con el mismo valor (para los *checkbox* y los *radio*, para lo *select* no es necesario).

· En las respuestas posibles que ofrecemos por cada pregunta, para indicar que una respuesta es la correcta, tan solo habrá que añadirle el atributo *value="bien"*, mientras que para indicar que la respuesta es errónea, habrá que poner el atributo *value="mal"*. Si la pregunta es multirespuesta (tipo *checkbok*) podrá haber varias respuestas correctas, pero si la pregunta es monorespuesta (tipo *radio*), solo podrá haber marcada una respuesta como buena. Los *select* funcionan como preguntas monorespuestas del tipo *radio*.

```
        <!-- ********** PREGUNTA 4 ********** -->
        <p/>
        <span class="italica">
        Pregunta 4: Indique cu&aacute;les de las siguientes afirmaciones son correctas
        </span>

        <br/>    
        <input type="checkbox" name="pregunta4" value="bien"/>
        La DCT y la DFT son transformadas separables.

        <br/>    
        <input type="checkbox" name="pregunta4" value="mal"/>
        La FFT de una imagen real es real.

        <br/>    
        <input type="checkbox" name="pregunta4" value="bien"/>
        La KLT es una transformada ortogonal.

        <br/>    
        <input type="checkbox" name="pregunta4" value="mal"/>
        La DCT por bloques de una imagen produce id&eacute;ntico resultado que
        la DCT de toda la imagen.
<p/>
```

· Poner en la etiqueta <form> de la página, el número de preguntas que tiene nuestra autoevaluación. Este dato es imprescindible para que el script calcule la nota sobre diez obtenida tras rellenarlo.

```
<!-- poner en el onSubmit, en el segundo parametro del metodo averiguarNota,
el numero de preguntas que hay en el formulario -->
<!--Si esta autoevaluación tiene 4 preguntas -->
<form name="formulario" onsubmit="averiguarNota(this,4);" action="">
```

- Modificar el /Autoevaluaciones/Indice/Indice.html para que incluya el enlace a la nueva autoevaluación, dentro del recuadro del curso al que pertenezca dicha autoevaluación.

4.1.4.4. Crear una autoevaluación nueva de un curso nuevo

Para crear una autoevaluación nueva de un curso nuevo:

- Crearte un directorio con el nombre del nuevo curso dentro de /Autoevaluaciones/ (recomendamos copiar uno de los existentes y renombrarlo). Ej. Autoevaluaciones/*AutoevNuevoCurso*/.

- Modificar el /Autoevaluaciones/Indice/Indice.html para que incluya un nuevo apartado con el nuevo curso y el enlace a la nueva autoevaluación de dicho curso.

[El resto del proceso de creación de la autoevaluación es similar al apartado anterior donde explicábamos como añadir una nueva autoevaluación a un curso ya existente.]

4.2. APPLETS

4.2.1. APPLETS: MEJORAS INTRODUCIDAS RESPECTO A VERSIONES ANTERIORES

Los applets de la aplicación están desarrollados en lenguaje de programación java, conformando un proyecto común en su conjunto.

Antes de esta revisión, dicho proyecto estaba formado por una intrincada red de clases sin ningún tipo de organización que diese coherencia estructural al mismo. En muchos casos, ni siquiera los nombres de las mismas eran identificativos de su funcionalidad o del applet al que pertenecían. La sucesión de distintos proyectos que han ido ampliando la aplicación sin duda no han contribuido de forma positiva al seguimiento de una línea común en el desarrollo de IMAGine.

Estas eran el conjunto de clases que formaban el proyecto de IMAGine antes de esta revisión:

AlgoritmosApplet.java	FuncionTransferencia.java	OppenApplet.java
AlgoritmosPanel.java	GiroPanel.java	PhaseApplet.java
AveragePanel.java	HistCanvas.java	ProcMorf.java
BaseImagenesPanel.java	HitMissApplet.java	PromedioApplet.java
CargaImagenPanel.java	HitMissPanel.java	QSortAlgorithm.java
CompFrecApplet.java	ImageBase.java	RuidoImpulPanel.java
Editor.java	ImageCanvas.java	RuidoPanel.java
EE.java	Imagen.java	RuidosoApplet_opp.java
EEOperMorfGrisesPanel.java	ImagenCanvas.java	RuidosoApplet_pro.java
EEOperMorfPanel.java	ImageSelectionPanel.java	TallerApplet.java
FFTImagen.java	ImageSelector.java	TFApplet.java
FFTImagenn.java	LogRuidoApplet.java	TFDeltasCanvas.java
FFTImagenold.java	Mascara.java	TFPanel.java
FiltroFrec.java	MascaraApplet.java	ThumbnailScroll.java
FiltroFrecPanel.java	miApplet.java	TransferenciaApplet.java
FiltroMascaraPanel.java	Observado.java	TransferenciaCanvas.java
FiltroMedianaPanel.java	OperMorfApplet.java	TTDIApplet.java
FrecuenciaApplet.java	OperMorfGrisesApplet.java	

Figura 4.14: Clases de IMAGine antes de este proyecto.

Muchas de ellas ni siquiera se utilizaban (FFTImagenn.java, FFTImagenold.java, LogRuidoApplet.java, RuidosoApplet_opp.java, RuidosoApplet_pro.java).

Sobre el resto de clases, se necesitaba revisar su funcionalidad, agruparlas por applets para unificar la estructura interna de aquellas que hiciesen la misma funcionalidad (por ejemplo las que heredan de Applet) y revisar la definición de elementos *private*/*protected*/*public* en función del uso real que se hiciese en el proyecto (y no definir todo como *public* por regla general). Si se hace una clase que hereda de Panel independiente para crear un panel del applet, parece lógico que esa clase tenga los métodos get/set para obtener los elementos seleccionados o introducidos en ese panel. Estos son solo ejemplos de la revisión tanto estructural como funcional que se ha llevado a cabo en esta revisión.

Intentando establecer una organización coherente para esclarecer su uso real, agrupando las clases por applets y funcionalidad, se llega a este esquema:

▪ **Clases referentes a los applets:**

<table>
<tr><th>NOMBRE APPLET</th><th colspan="2">Interfaz gráfica</th><th>Ejecución Applet</th><th colspan="2">Funcionamiento</th></tr>
<tr><td>‘Básico’, ‘Filtro de mediana’ y ‘Rotación de imágenes’</td><td rowspan="10">ImagenCanvas.java*</td><td>-</td><td>TallerApplet.java</td><td>(Opciones del submenú de Imagen)</td><td rowspan="10">Imagen.java**</td></tr>
<tr><td>‘Operaciones puntuales’</td><td>TransferenciaCanvas.java,</td><td>TransferenciaApplet.java

Observado.java</td><td>FuncionTransferencia.java</td></tr>
<tr><td>‘Filtrado máscaras’</td><td>FiltroMascaraPanel.java,</td><td>MascaraApplet.java</td><td>(Opciones del submenú de Imagen)

Mascara.java</td></tr>
<tr><td>‘Importancia componentes baja frecuencia’</td><td>-</td><td>CompFrecApplet.java</td><td>FiltroFrec.java</td><td rowspan="4">FFTImagen.java</td></tr>
<tr><td>‘Contribución componentes frecuenciales en la DFT’</td><td>TFDeltaCanvas.java</td><td>TFApplet.java</td><td>-</td></tr>
<tr><td>‘Importancia de la fase’</td><td>-</td><td>PhaseApplet.java</td><td>-</td></tr>
<tr><td>‘Experimento del Oppenheim</td><td>-</td><td>OppenApplet.java</td><td>-</td></tr>
<tr><td>‘Promedio de módulos y fases’</td><td>AveragePanel.java</td><td>PromedioApplet.java</td><td>-</td></tr>
<tr><td>‘Operaciones morfológicas binarias’</td><td>EEOperMorfPanel.java</td><td>OperMorfApplet.java</td><td rowspan="4">ProcMorf.java, EE.java</td></tr>
<tr><td>‘Hit or miss’</td><td>HitMissPanel.java</td><td>HitMissApplet.java</td></tr>
<tr><td>‘Algoritmos morfológicos’</td><td>AlgoritmosPanel.java</td><td>AlgoritmosApplet.java</td></tr>
<tr><td>‘Operaciones morfológicas escala grises’</td><td>EEOperMorfGrisesPanel.java</td><td>OperMorfGrisesApplet.java</td></tr>
</table>

* ImagenCanvas.java es la clase que se utiliza como contenedor para mostrar cualquier imagen en un applet, por eso todos los applets implementan dicha clase para desarrollar su interfaz (ya que todos muestran imágenes). Incluso hay applets que solo usan esa clase para crear su interfaz sin necesidad de ninguna clase más auxiliar, puesto que solo muestran imágenes (sson los applets en los que aparece un guión en el apartado ‘Interfaz’). También es la encargada de desarrollar el interfaz necesario para visualizar el submenú desplegable.

** Imagen.java es la clase que implementa el funcionamiento de las opciones del submenú que aparece al pinchar con el botón derecho del ratón sobre cualquier imagen. Como a lo largo de todos los applets de IMAGine este submenú siempre va a estar disponible al pinchar sobre cualquier imagen que se visualice en ellos, consideramos que todos los applets implementan esta clase para su funcionamiento. Hay applets cuya finalidad específica se encuentre en una de las propias opciones de este submenú (por ejemplo el de ‘Filtrado máscaras’). En esos casos, en el apartado ‘Funcionamiento’ de la tabla aparecerá la indicación ‘Opciones del submenú de Imagen’ (esto es porque la base del funcionamiento de esta opción está implementado en la propia clase Imagen.java).

▪ **Clases referentes al submenú desplegable de Imagen (los paneles emergentes de algunas opciones):**

· Transformada de Fourier: TFPanel.java
· Mostrar Histograma: HistCanvas.java
· Filtrado Máscara: FiltroMascaraPanel.java
· Filtrado Mediana: FiltroMedianaPanel.java
· Filtrado Frecuencia: FiltroFrecPanel.java, FFTImagen
· Girar Imagen: GiroPanel.java
· Añadir Ruido Gaussiano: RuidoPanel.java
· Añadir Ruido Impulsivo: RuidoImpulPanel.java
· Cargar Imagen: CargaImagenPanel.java, etc.

▪ **Clases para el Editor de Imágenes (Editor que permite modificar las imágenes cargadas adaptándolas para su uso en los applets de IMAGine):**

· Clase que arranca la aplicación gráfica del Editor: Editor.java
· Resto de clases: BaseImagenesPanel, CargaImagenPanel, ImageBase, ThumbnailScroll, ImageSelectionPanel, ImageSelector.

Figura 4.15: Estructura clases de IMAGine antes de este proyecto.

Esta agrupación, como se puede apreciar, no es nada intuitiva inicialmente puesto que los nombres de las clases por si mismos no indican ninguna agrupación, y tampoco hay ninguna estructura de carpetas, paquetes, etc. que suplan esta carencia. Hemos intentado estructurar IMAGine para que el resultado tenga una organización lo suficientemente intuitiva como para que una agrupación de estas características sea algo inmediato. Con esto intentamos que, de cara a futuras ampliaciones, no haya que pasar por un análisis previo de estas características para comprender el funcionamiento de la aplicación.

Por tanto, el primer tema que hemos abordado ha sido la estructuración del proyecto en packages para agrupar las clases por applets y funcionalidad.

Hemos realizado otras mejoras significativas, como el cambio de nombre de algunas clases para que éste fuese más identificativo de la funcionalidad de la misma. Hemos suprimido las clases que no se usaban y agrupado otras, como las que implementaban los paneles emergentes del submenú. Hemos creado una clase interfaz de cada applet con la definición de constantes que se utilizan en varias clases del mismo para crear así una estructura más transparente e intuitiva.

Nótese que el número de clases ha aumentado también porque está incluida la funcionalidad de imágenes a color (incluyendo el Editor de Imágenes para añadir nuevas imágenes a color a la base de imágenes que utilizan los applets de este tipo) y la creación de los applets del curso de restauración de imágenes.

Figura 4.16: Estructura clases de IMAGine en este proyecto.

Finalmente, pasamos a resumir a grandes rasgos las mejoras introducidas, ya que entraremos en detalle sobre ellas en el análisis técnico.

- Eliminación de clases obsoletas que no se utilizaban

- Nombres identificativos para las clases

- Agrupación de todas las clases del proyecto en packages juntándolas por funcionalidad y applet al que pertenecían.

- Agrupación de clases que implementaban paneles emergentes de los submenús en una sola, para eliminar así el exceso de clases innecesarias.

- Creación de clases de tipo interface para la definición de ctes en los applets, logrando así una estructura más transparente en ellos.

- Reestructuración interna de las clases que heredan de Applet para que todas sigan una estructura idéntica.

- Clases que heredan de Panel con métodos set/get para obtener los valores introducidos o seleccionados en el propio panel.

- Revisión de la definición de elementos como *private*/*public* según correspondía en realidad (y no definición de todos como *public* como regla general).

- Reestructuración interna de los métodos de la clase Imagen agrupándolos por funcionalidad y cursos.

- Corrección de pequeños errores funcionales de los applets del curso Procesado Morfológico (borrado de los contenedores al cargar una nueva imagen en el contenedor principal), así como otros pequeños detalles.

- Creación de los applets del curso Restauración de Imágenes siguiendo todas las pautas definidas en estos apartados.

- Creación de la funcionalidad de Imagen a color para trabajar en los applets no solo con imágenes a escala de grises. Además, las imágenes a color que se utilicen conservarán su tamaño original, no siendo necesario recortarlas ni editarlas para su uso en los applets, como sí ocurría con las imágenes en escala de grises de la clase Imagen. Implementación de las técnicas principales de procesado (filtrado, FFT, etc.) para ellas.

- Editor de imágenes:

· En imágenes grises: carga con extensión .jpg y documento .txt con el listado de imágenes. Antes se cargaban serializadas con una extensión que no permitía visualizarlas con una aplicación de imágenes normal, lo que complicaba el mantenimiento de esta base de imágenes. Ahora, al no estar serializadas las imágenes ni el documento que contiene el listado de las mismas, es más fácil e intuitivo mantener la base de imágenes. Además, se han añadido paneles emergentes informativos con el resultado de la carga/borrado de imágenes.

· En imágenes grises: Eliminación de imágenes de las FFT de las imágenes precargadas en el servidor. Ya no se generan al cargar una nueva imagen. Se calculan dinámicamente al seleccionar la opción fft (la velocidad actual de los procesadores hace que no sea necesario tener las imágenes precalculadas ya que se pueden generar dinámicamente).

· En imágenes a color: creación de toda la funcionalidad de carga de imágenes a color (conservando en la carga el tamaño original, creando además la imagen en miniatura (generarla dinámicamente resultaba muy costoso en cuanto a tiempo de generación en el servidor, por lo que se optó por precargarla), creación del listado de imágenes para poder acceder a ellas…). Por supuesto, la carga de estas imágenes también la hacemos sin serializar (las cargamos simplemente con extensión .jpg).

4.2.2. APPLETS: FUNCIONAMIENTO A NIVEL DE USUARIO

IMAGine consta de una serie de applets especialmente pensados para dotar el curso con una herramienta interactiva donde el alumno pueda practicar los conocimientos explicados en la teoría de una manera dinámica, fácil e intuitiva.

En esta sección explicaremos el funcionamiento básico de todos ellos, haciendo especial hincapié en los del curso de Restauración de imágenes (ya que estos son los applets creados en este proyecto, los otros ya existían en versiones anteriores de IMAGine).

4.2.2.1. APPLETS DEL CURSO 'PROCESADO BÁSICO DE IMG'

4.2.2.1.1. Applet Básico de Imágenes a escala de grises

Este applet se encuentra en la *portada* de IMAGine, en la sección *Funcionamiento del curso*. En él simplemente aparece una imagen sobre la que, pulsando el botón derecho del ratón, se despliega un menú donde aparecen las distintas operaciones que se le pueden realizar a dicha imagen. Este menú desplegable estará siempre presente en el resto de los applets de IMAGine al pinchar con el botón derecho del ratón sobre una imagen en escala de grises.

Las operaciones que incluye son:

- Aumentar la imagen 4X (Zoom de la imagen).
- Realizar su transformada de Fourier (Módulo y fase).
- Mostrar el histograma con los valores de las diferentes componentes de gris que posee la imagen.
- Invertir los tonos de grises de la imagen (negativo).
- Ecualizar el histograma.
- Filtrar la imagen con una máscara.
- Filtrar la imagen con un filtro de mediana.
- Filtrar la imagen en frecuencia (Paso-alto o paso-bajo, ideal o de Butterworth).
- Girar la imagen un cierto ángulo.
- Añadir ruido gaussiano.
- Añadir ruido impulsivo.
- Cargar una imagen de la base de imágenes del proyecto.

Figura 4.17: Menú desplegable al pinchar con el botón derecho sobre una imagen en escala de grises.

4.2.2.1.2. Applet "Operaciones Puntuales"

Este applet permite observar los cambios que sufre una imagen cuando cambias la intensidad de cada uno de sus puntos individuales, utilizando para ello una función de transferencia.

Se puede practicar la aplicación de distintas funciones de transferencia, como las de tipo *recorte*, *umbralización*, *negativo*… y todas aquellas que se le ocurran al alumno, puesto que el panel superior del applet muestra una función de transferencia que se puede modificar añadiendo tantos puntos como se desee.

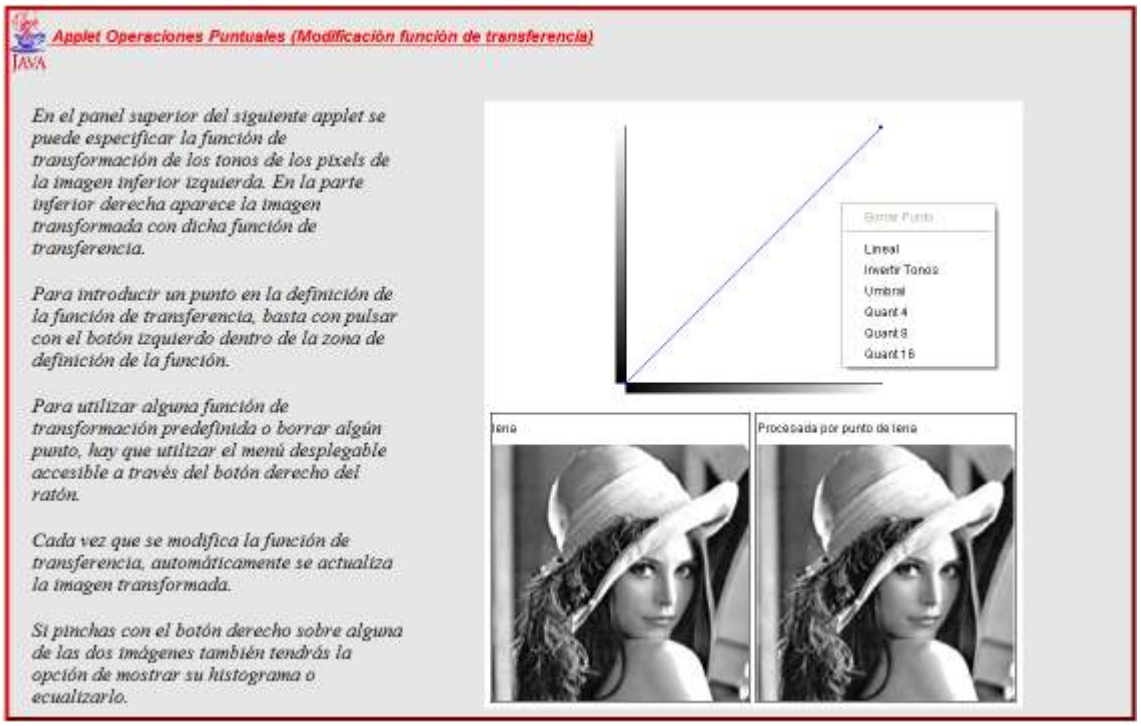


Figura 4.18: Applet Operaciones Puntuales.

Veamos un ejemplo de modificación del contraste de una imagen utilizando una función de transferencia dada:

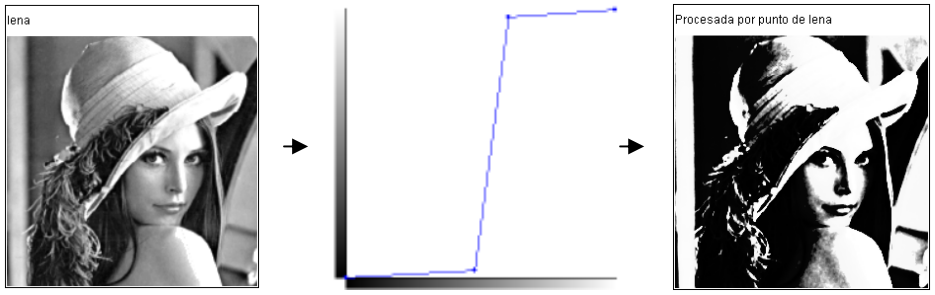


Figura 4.19 : Ejemplo de operación puntual (modificación del contraste).

4.2.2.1.3. Applet "Filtrado Basado en Máscaras"

Este applet nos permite practicar los efectos del filtrado basado en máscaras. Para ello podemos utilizar una máscara predefinida, seleccionándola en la pestaña que aparece en la parte superior del applet (Paso Bajo, Paso Alto, Laplaciano…), o bien definir una de manera manual marcando los valores en las casillas que aparecen inmediatamente debajo de la pestaña que nombrábamos anteriormente (se pueden poner los valores de la máscara y un denominador para dividir todos esos valores por él).

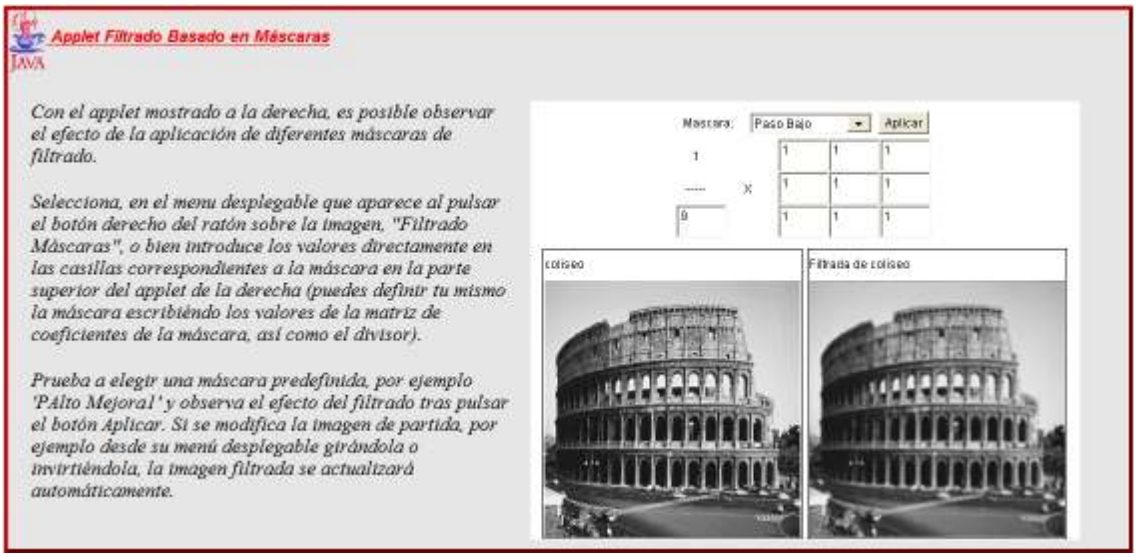


Figura 4.20: Applet Filtrado basado en máscaras.

4.2.2.1.4. Applet "Importancia Componentes de Baja Frecuencia"

Las componentes de baja frecuencia tienen gran importancia a la hora de definir imágenes. Con este applet el alumno puede apreciar este hecho. Para ello, se muestran dos imágenes, la de la izquierda corresponde a la imagen original y la de la derecha es la imagen resultante de filtrar la imagen original con un filtro paso bajo ideal de distinta frecuencia de corte. A medida que la frecuencia de corte del filtro paso bajo ideal va aumentando (la frecuencia de corte va aumentando cada vez que pulsamos el botón *continuar*) la imagen se va haciendo más nítida, hasta llegar a ser casi idéntica a la original (en ese momento la imagen llega a emplear 128 coeficientes).

Figura 4.21: Applet Importancia componentes de baja frecuencia.

Con el siguiente ejemplo (figura) vemos como la imagen se va haciendo más nítida cada vez que pulsamos el botón *continuar* del applet. Esto es debido a lo que explicábamos antes: la imagen resultante de filtrar la imagen original con el filtro paso bajo ideal de frecuencia de corte cada vez mayor emplea más coeficientes y por tanto se ve mejor.

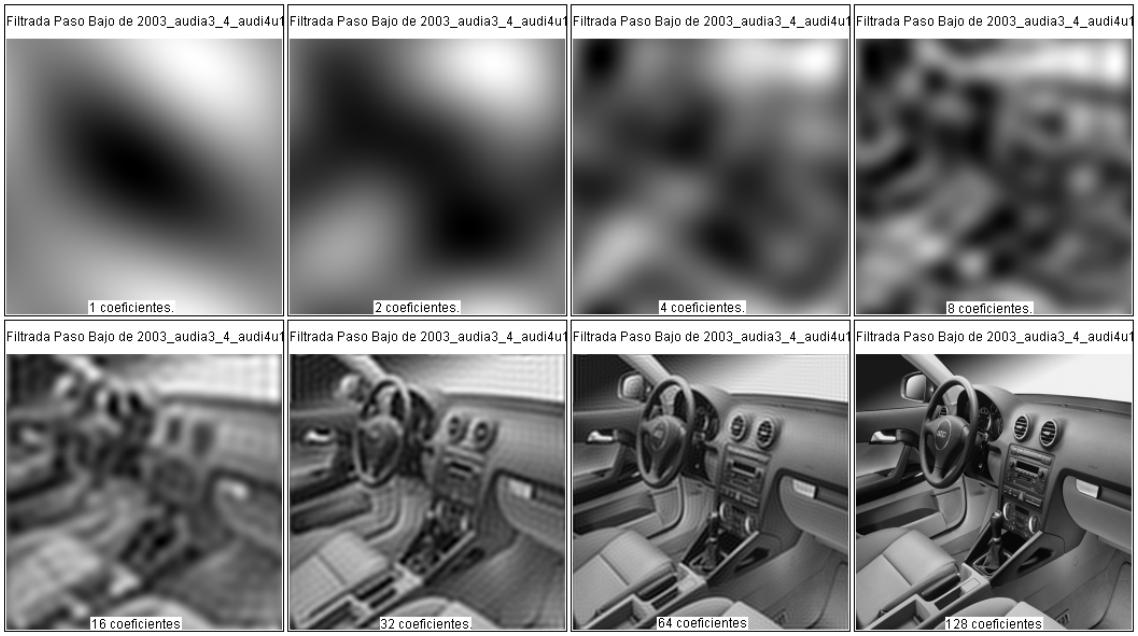


Figura 4.22: Ejemplo de importancia de las componentes de baja frecuencia

4.2.2.1.5. Applet "Filtro de Mediana"

En este applet solo mostramos una imagen (*applet básico*), sobre la que proponemos añadir ruido impulsivo o ruido gaussiano, para filtrarla después con un filtro de mediana y observar los resultados.

El filtro de mediana es el que mejor funciona para eliminar el ruido impulsivo, ya que preserva más los bordes que si filtrásemos una imagen con dicho ruido con un filtro paso-bajo. El filtro paso bajo difumina más los bordes ya que hace un promediado de los píxeles de la vecindad. Sin embargo, el filtro de mediana, para un punto dado, ordena todos los píxeles de la vecindad de menor a mayor, y a ese punto, le asigna el punto correspondiente a la mediana de esa lista ordenada. De esta forma se elimina el ruido impulsivo (el ruido suele aparecer como puntos blancos o negros, y al pasar el filtro de mediana, para un punto dado nunca se elegirán los valores correspondientes al ruido porque estos estarán en los extremos de la lista resultante al ordenar los puntos de la vecindad de menor a mayor).

Esto es lo que pretendemos que observe el alumno a través de este applet.

La opción de 'filtrado mediana' permite seleccionar el tamaño de la máscara que utilizaremos en el promediado de cada píxel de la imagen. A mayor tamaño, se perderá más nitidez en la imagen, pero a cambio conseguiremos eliminar con mayor eficacia las imágenes más ruidosas. El desplegable permite seleccionar máscaras de tamaño 3x3, 5x5 o 7x7.

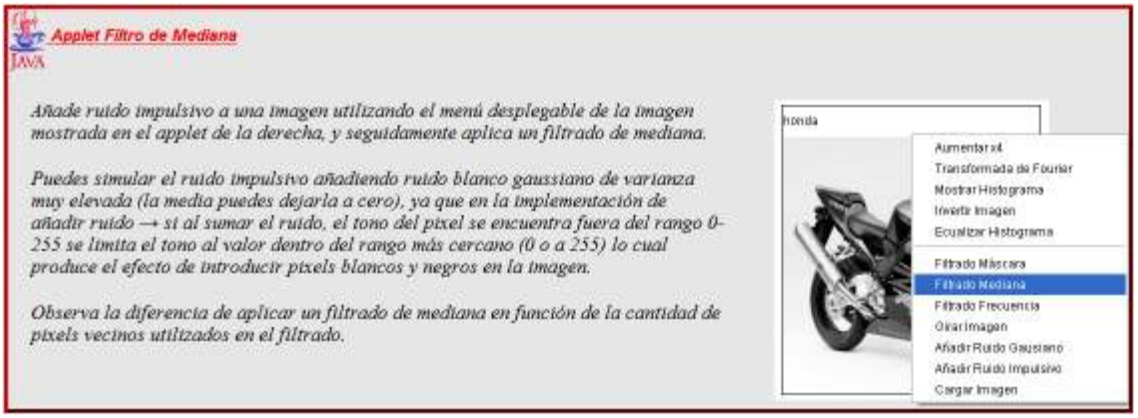


Figura 4.23: Applet Filtro de mediana.

Aquí se muestra un ejemplo de utilización del applet [figura]: a una imagen dada le añadimos ruido, en este caso gaussiano, y después filtramos la imagen resultante con un filtro de medianas para eliminar el efecto de ese ruido. Vemos como la imagen final pierde algo de nitidez respecto a la de partida, pero que en cambio en ella no aparece el ruido que veíamos en la imagen intermedia.

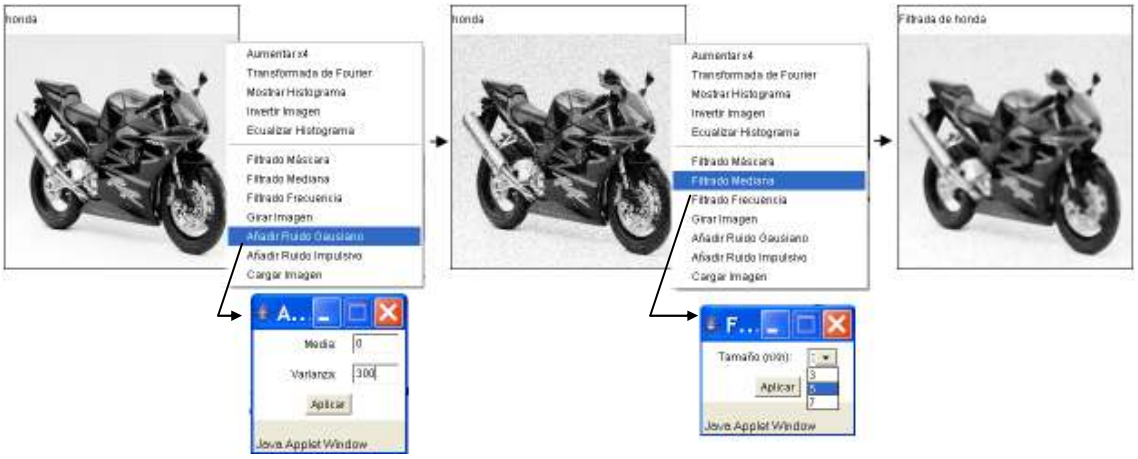


Figura 4.24: Ejemplo de la utilización del filtro de mediana.

4.2.2.1.6. Applet "Rotación de Imágenes y su efecto en la DFT"

Este applet simplemente pretende ilustrar el efecto que tiene sobre la DFT la rotación de imágenes.

La DFT tiene una propiedad denominada "Propiedad del desplazamiento circular" que dice que el módulo de la DFT es invariante frente a desplazamientos circulares. En este caso no estamos haciendo exactamente desplazamientos circulares, sino que estamos rotando la imagen, pero el efecto es parecido ya que la transformada de Fourier de la imagen girada es igual a la de la imagen original, pero girada tantos grados como ángulo hallamos girado la imagen original. Además, para no alterar la DFT al girar la imagen, al realizar la rotación, los píxeles que salen por un lado aparecen por el otro.

El alumno puede investigar de manera práctica como afecta la rotación de imágenes en el módulo y la fase de su transformada de Fourier.

Figura 4.25: Applet Rotación de imágenes y su efecto en la DFT

El procedimiento a seguir en este applet es el siguiente: mostrar la DFT de la imagen original (menú desplegable sobre la imagen), después rotar la imagen un determinado número de grados (también a través del menú desplegable seleccionando la opción de *Girar imagen*) y finalmente mostrar la DFT de la imagen girada. Se pretende comparar la DFT de la imagen original con la DFT de la imagen girada.

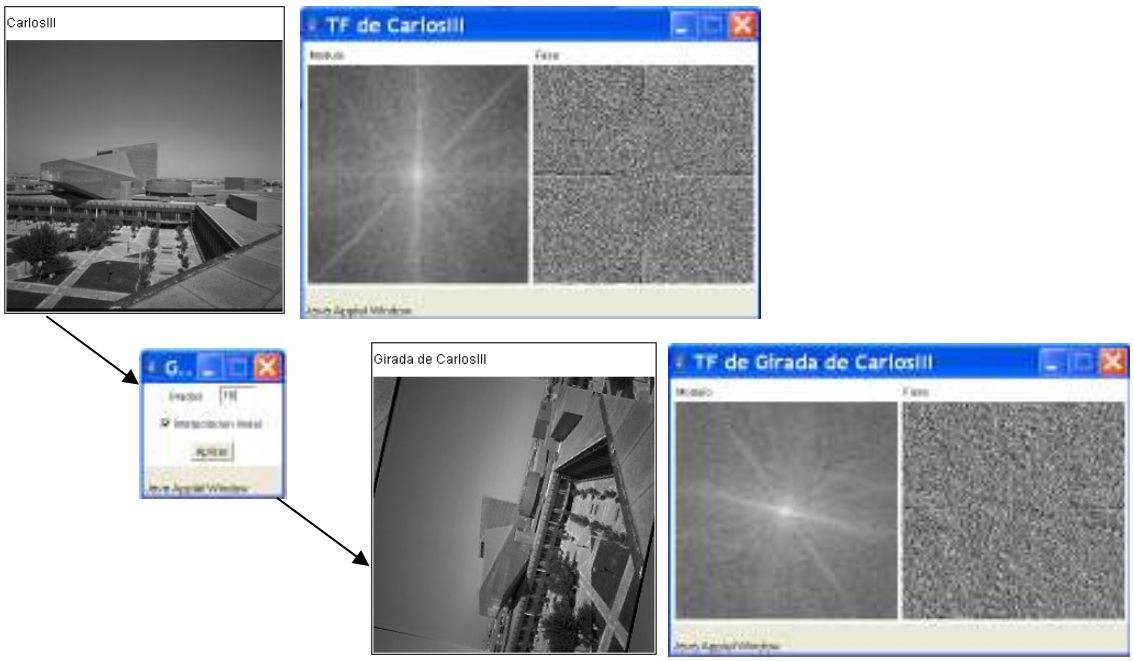


Figura 4.26: Ejemplo de rotación de imágenes y su efecto en la DFT

4.2.2.1.7. Applet “Contribución de las componentes frecuenciales en la DFT”

En este applet se pretende que el usuario observe el efecto que tiene las componentes de menor frecuencia en la composición de una imagen. Para ello aparece en la parte izquierda del applet un panel que contiene un plano de frecuencia donde se pueden colocar deltas y cambiar la fase de dichas deltas que hacen que aparezca en la parte derecha del applet la imagen cuya transformada se ha especificado en el plano de frecuencia.

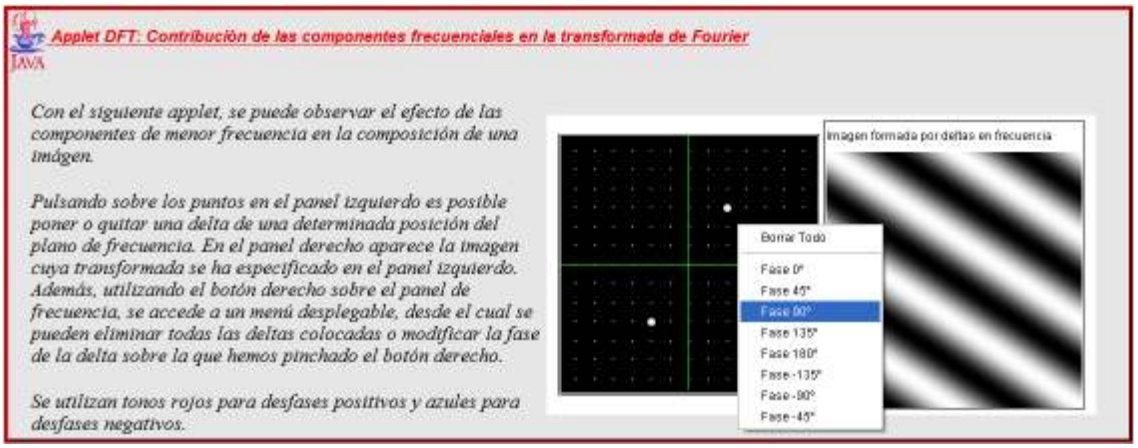


Figura 4.27: Applet Contribución de las componentes frecuenciales en la transformada de Fourier.

4.2.2.1.8. Applet “Importancia de la fase”

Los tres applets que vienen a continuación son experimentos basados en la Transformada de Fourier de imágenes. Para ello jugaremos con los módulos y fases de las imágenes, probando a intercambiarlos entre sí, poniéndolos a un valor nulo, etc.

Concretamente, en este applet pretendemos hacer un experimento que demuestre la importancia de la fase respecto al módulo. Para ello, simplemente cogemos una imagen (mostrada en la parte superior del panel), y a partir de ella calculamos dos nuevas imágenes:

- La imagen de la parte inferior izquierda está formada por la *fase* de la imagen original y *módulo* unidad.

- La imagen de la parte inferior derecha está formada por el *módulo* de la imagen original y *fase* cero.

Vemos como la imagen de la izquierda (la que conserva la fase de la imagen original) guarda algún parecido con la original, mientras que la de la derecha (la que conserva el módulo de la imagen original) apenas guarda ningún parecido, por no decir ninguno, con dicha imagen.

Con esto queda demostrado que la fase de una imagen cualquiera guarda más información sobre dicha imagen que su propio módulo.

Para hacer este experimento con cualquier otra imagen, solo basta con pinchar con el botón derecho sobre la imagen superior y seleccionar la opción ‘Cargar imagen’.

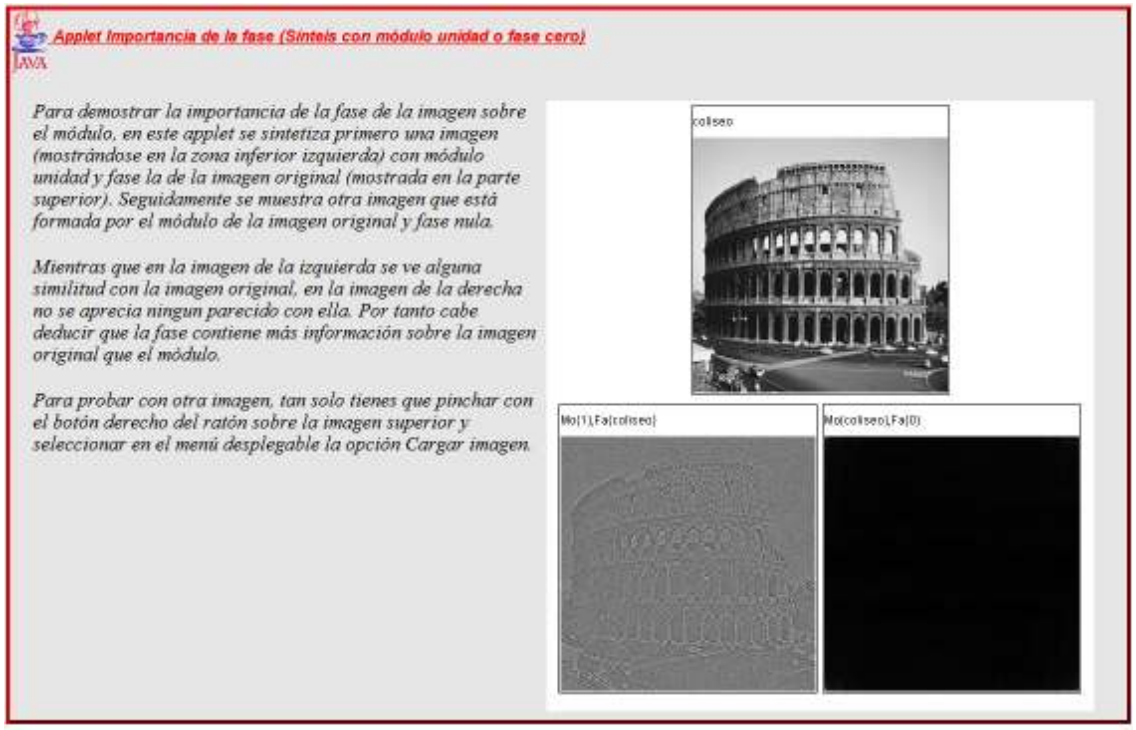


Figura 4.28: Applet Importancia de la fase (síntesis con módulo unidad o fase cero)

4.2.2.1.9. Applet "Experimento del Oppenheim"

Este es el segundo de los tres applets que constituyen los experimentos con los módulos y fases de la Transformada de Fourier de Imágenes.

Denominaremos este experimento como el Experimento del Oppenheim (puesto que viene sugerido en uno de los capítulos del libro Señales y Sistemas, más conocido como Oppenheim).

Este experimento consiste en, dadas dos imágenes, intercambiar entre ellas sus módulos. Las imágenes resultantes estarán formadas por el módulo de la imagen contraria y la fase de la imagen original.

Las imágenes originales son las que se muestran en la parte superior del applet, y las resultantes del experimento son las de la parte inferior.

Observamos como las imágenes obtenidas se parecen bastante a las originales, quizá con cierta distorsión provocada por el intercambio de módulos…pero en cualquier caso seguimos llegando a la misma conclusión que en el caso del applet anterior (Applet 'Importancia de la fase') → La fase de una imagen cualquiera contiene más información sobre dicha imagen que su módulo (pese a haber cambiado el módulo de la imagen, tan solo conservando la fase de la imagen original, se siguen observando los principales rasgos y trazos de la ya nombrada imagen original).

Para probar el experimento con cualquier otro par de imágenes, seguimos el procedimiento habitual → se pincha con el botón derecho sobre las imágenes mostradas en la parte superior del applet y en el menú desplegable seleccionamos la opción 'Cargar imagen'.

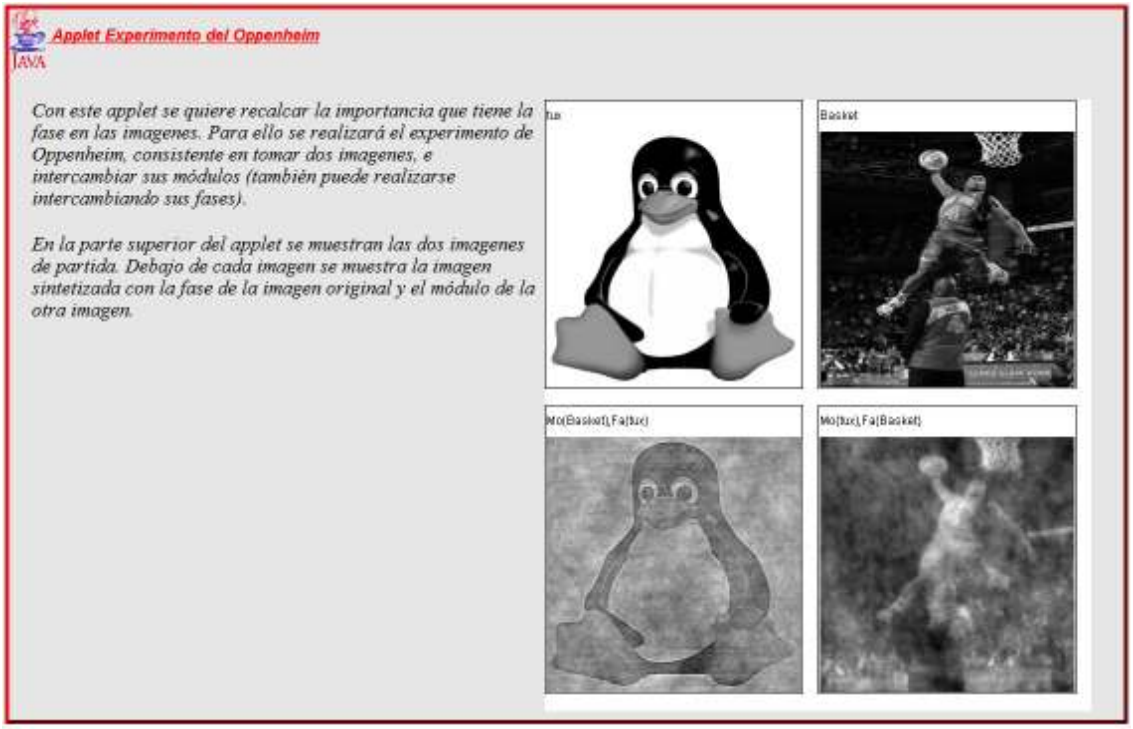


Figura 4.29: Applet Experimento del Oppenheim

4.2.2.1.10. Applet "Promedio de módulos y fases"

El último de los tres experimentos basados en la Transformada de Fourier de imágenes, muestra una vez más la importancia de la fase con respecto al módulo.

Para demostrarlo, experimentaremos de nuevo con dos imágenes (la primera se muestra en la parte superior del applet y la segunda se selecciona pinchando en el botón 'Elegir imágenes promedio'). Las nuevas imágenes obtenidas mediante el experimento son:

- La imagen inferior izquierda está construida por el *módulo* promedio de las dos imágenes originales y la *fase* de la primera imagen original.

- La imagen inferior derecha está formada por el *módulo* de la primera imagen original y la *fase* promedio de las dos imágenes originales.

Al utilizar para complementar el experimento, a diferencia de los dos applets anteriores, el promedio de dos imágenes para construir las imágenes nuevas [en vez de utilizar módulo unidad o fase cero como en el Applet 'Importancia de la fase'] tan solo estamos añadiendo un detalle más al experimento, pero la esencia es la misma en ambos casos: resaltar la importancia de la fase con respecto al módulo. Utilizando el promedio tan solo distorsionamos ligeramente las imágenes obtenidas al final del experimento.

De las dos imágenes calculadas, la que guarda más parecido con la primera imagen original es la que utiliza la fase de dicha imagen (correspondiente con la imagen inferior izquierda). En cambio, la imagen construida con el módulo de la primera imagen original apenas guarda similitud (visualmente hablando) con esa primera imagen original.

Con este ejemplo, como apuntábamos en los párrafos anteriores, queda de nuevo patente la importancia de la fase con respecto al módulo.

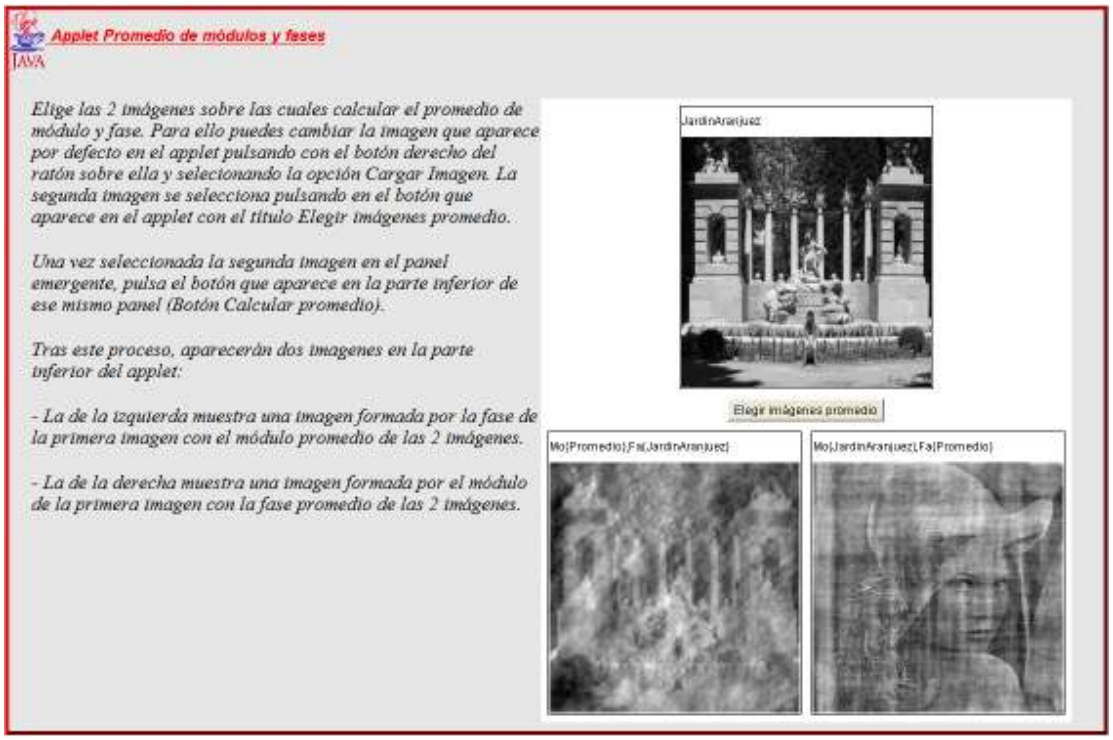


Figura 4.30: Applet Promedio de módulos y fases

4.2.2.2. APPLETS DEL CURSO 'PROCESADO MORFOLÓGICO'

Los applets del curso procesado morfológico contienen las técnicas más habituales en este tipo de procesado (erosión, dilatación, apertura o cierre), así como los algoritmos más utilizados en procesado morfológico (hit or miss, adelgazamiento, engorde…).

El curso no solo se limita a las imágenes en binario, más sencillas de tratar en el procesado morfológico, sino que también se extiende a las imágenes en escala de grises.

4.2.2.2.1. Applet "Operaciones morfológicas sobre imágenes binarias"

Con este applet se pueden practicar los efectos de los operadores morfológicos sobre las imágenes.

En primer lugar tenemos que elegir la operación morfológica que queremos realizar sobre la imagen (dilatación, erosión, apertura o cierre).

Posteriormente seleccionaremos el elemento estructurante que utilizaremos en la operación morfológica. Este puede ser un EE predefinido (seleccionándolo en la lista que aparece en el segundo desplegable) o bien uno diseñado por el propio usuario (para ello el usuario deberá dibujarlo en el panel que se encuentra en la esquina superior derecha del applet).

Llegado a este punto es el momento de realizar la operación morfológica pinchando en el botón 'Aplicar a la imagen original'. La imagen de la izquierda será la imagen binarizada de la imagen original, mientras que la imagen de la derecha será la imagen resultante de aplicar la operación morfológica seleccionada. En sucesivas iteraciones tendremos la opción de elegir si aplicar la operación a la imagen original o a la imagen resultante de la anterior operación morfológica (pinchando en este caso en el botón 'Aplicar al resultado').

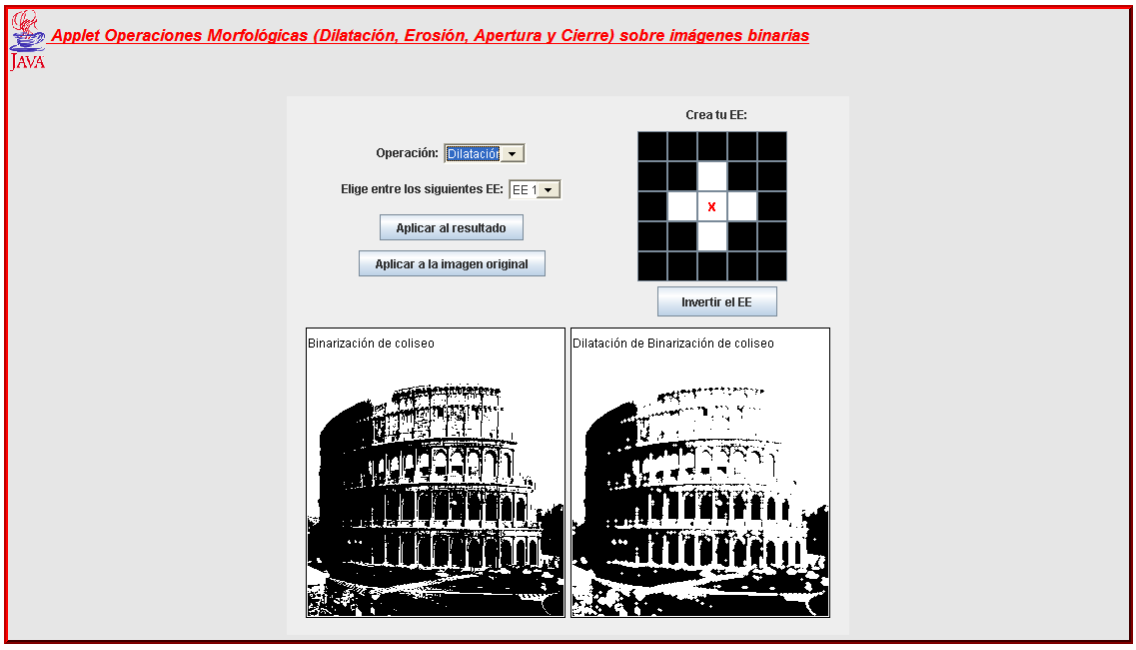


Figura 4.31: Applet Operaciones morfológicas sobre imágenes binarias.

4.2.2.2.2. Applet "Transformación Hit or Miss (Acierta/Falla)"

El algoritmo Hit or Miss es fruto de una combinación de operaciones morfológicas en cadena, pero debido a su gran utilización en la mayoría de algoritmos morfológicos, creemos conveniente dedicarle un apartado.

El algoritmo Hit or Miss trata de localizar un patrón dado en una imagen (un EE de determinada forma con puntos blancos y negros → un objeto rodeado de un fondo). El resultado del algoritmo nos dará los puntos de la imagen donde se da ese patrón (los puntos donde, colocado el centro del patrón o EE, coincide exactamente con la imagen original).

Veamos el siguiente ejemplo (figura) y en base a él explicaremos el algoritmo Hit or Miss:

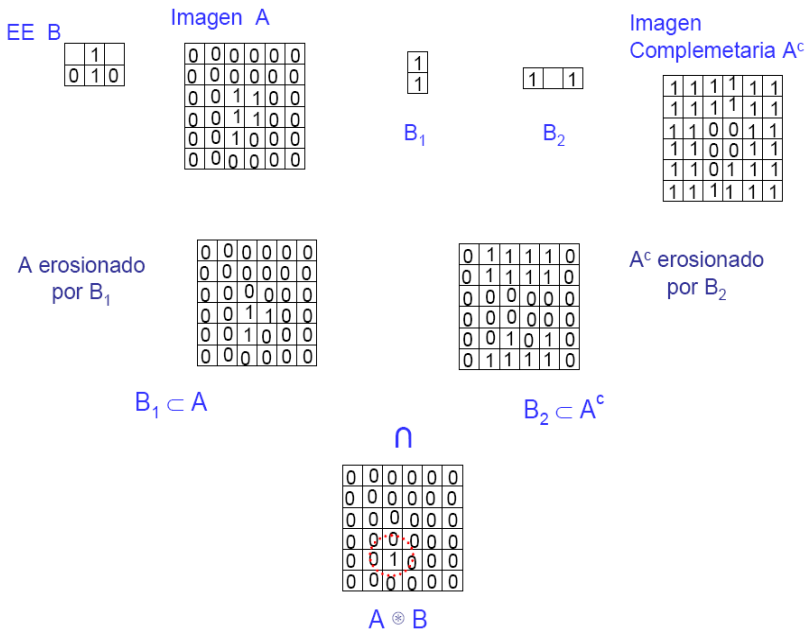


Figura 4.32: Algoritmo Hit or Miss.

Queremos localizar el EE B [cuyo objeto (unos en la figura) lo definimos como B_1 y su fondo (ceros en la figura) como B_2] en la imagen A:

- Para ello erosionamos A con B_1 obteniendo los puntos de la imagen donde hay un patrón similar al 'objeto' del EE B (ya que al erosionar la imagen solo se colorearán los puntos donde el EE B_1 quepa completamente dentro del objeto de la imagen A).

- Por otro lado vamos a hacer lo mismo para localizar los puntos donde el 'fondo' de EE B coincide con el 'fondo' de la imagen original A: Para ello hallamos la imagen complementaria de la imagen original A y por otro lado el complementario al fondo del EE B que denominaremos B_2. Erosionamos A complementaria con B_2 obteniendo los puntos donde ambos coinciden.

Finalmente hacemos la intersección entre las dos imágenes halladas por separado obteniendo una imagen resultante donde aparecerá un 1 en los puntos donde, situado el centro del EE B, 'objeto' y 'fondo' del EE B coincidan con 'objeto' y 'fondo' de la imagen original A.

Con este applet podemos practicar el uso de este interesante algoritmo morfológico. Para ello primero seleccionamos el EE, que al igual que en el applet anterior, podemos crearlo nosotros mismos o elegirlo en el desplegable que aparece en la parte superior del applet (también tenemos el botón 'Invertir el EE' por si queremos cambiar 'objeto' por 'fondo' y viceversa → cambiar 1 a 0 y a la inversa). Después tenemos que pulsar el botón 'Aplicar a al imagen original' para ejecutar el algoritmo (en sucesivas iteraciones también tendremos la posibilidad de aplicar el algoritmo al resultado de la operación anterior).

La imagen de la izquierda será la binarización de la imagen original, mientras que la de la derecha será el resultado de aplicar Hit or Miss.

Recordemos que pinchando con el botón derecho del ratón sobre cualquier imagen del curso podremos cargar cualquier otra imagen para experimentar en los applets con ella.

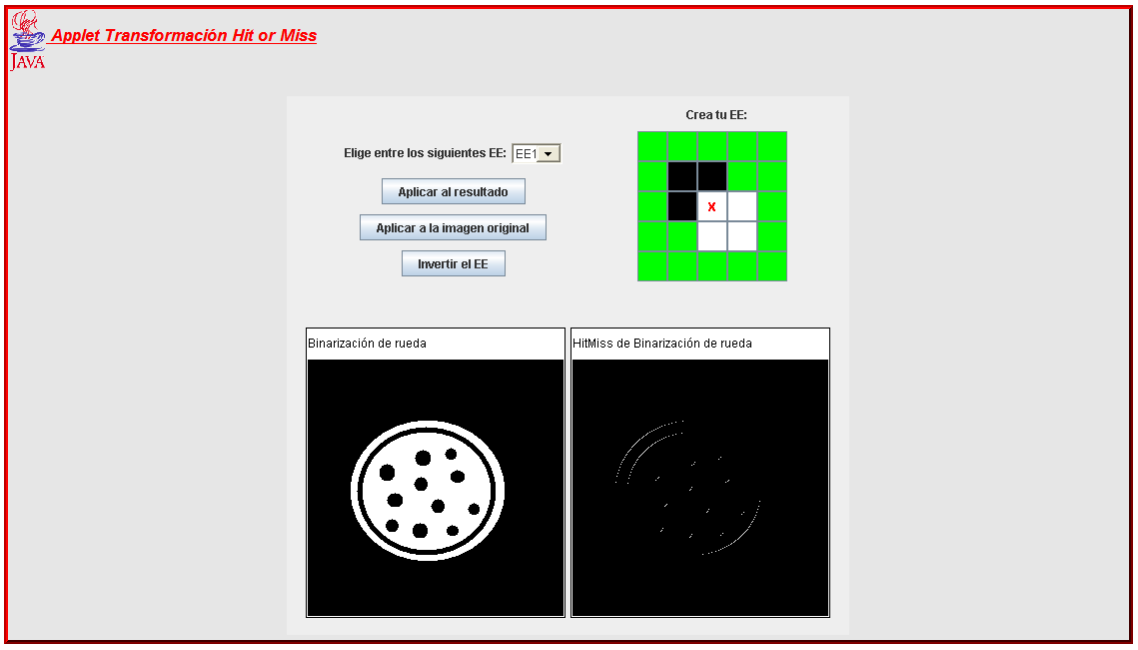


Figura 4.33: Applet Transformación Hit or Miss.

4.2.2.2.3. Applet “Algoritmos morfológicos”

A partir de este applet se pueden practicar los algoritmos morfológicos más importantes en el procesado de imágenes (borde exterior/interior/ancho, adelgazamiento/engorde, esqueleto, cerco convexo y poda).

Podemos elegir el EE con el que ejecutar el algoritmo seleccionado entre los predefinidos o creando uno propio.

A la hora de ejecutar el algoritmo tenemos varias opciones. ‘Aplicar a la imagen original’ implica que el algoritmo se aplica sobre la imagen original cargada en la esquina inferior izquierda (mostrándose el resultado en la imagen de la esquina inferior derecha), mientras que ‘Aplicar al resultado’ tomará como imagen de partida para aplicar el algoritmo la que haya en ese momento en la esquina inferior derecha (fruto de ejecuciones anteriores).

Con la finalidad de aumentar el valor didáctico de este applet, se incluye la opción de ‘Aplicar paso a paso’, que consiste en ir mostrando los pasos que sigue el algoritmo para conseguir el resultado deseado. Al pulsar sobre este botón, aparecerá inmediatamente un botón debajo (coloreado en rojo) que describirá el siguiente paso que realizará el algoritmo. Por ejemplo, el algoritmo ‘Borde exterior’ consiste en hacer en primer lugar una erosión de la imagen original binarizada (aparecerá un botón que ponga: ‘Paso 1: Erosión’) y seguidamente restar a la imagen original binarizada el resultado de esa erosión (en este caso aparecerá en el botón la leyenda ‘Paso 2: Original-Erosión). Así el alumno podrá comprender mejor como funcionan cada uno de los algoritmos morfológicos principales que se explican en la teoría y ver los resultados de su aplicación paso a paso.

Por último, dado que hay algoritmos bastante complejos que tardan algunos segundos es ejecutarse, se incluye a modo informativo una etiqueta que muestra los segundos consumidos durante la ejecución del algoritmo, y al terminar el proceso muestra el tiempo total empleado.

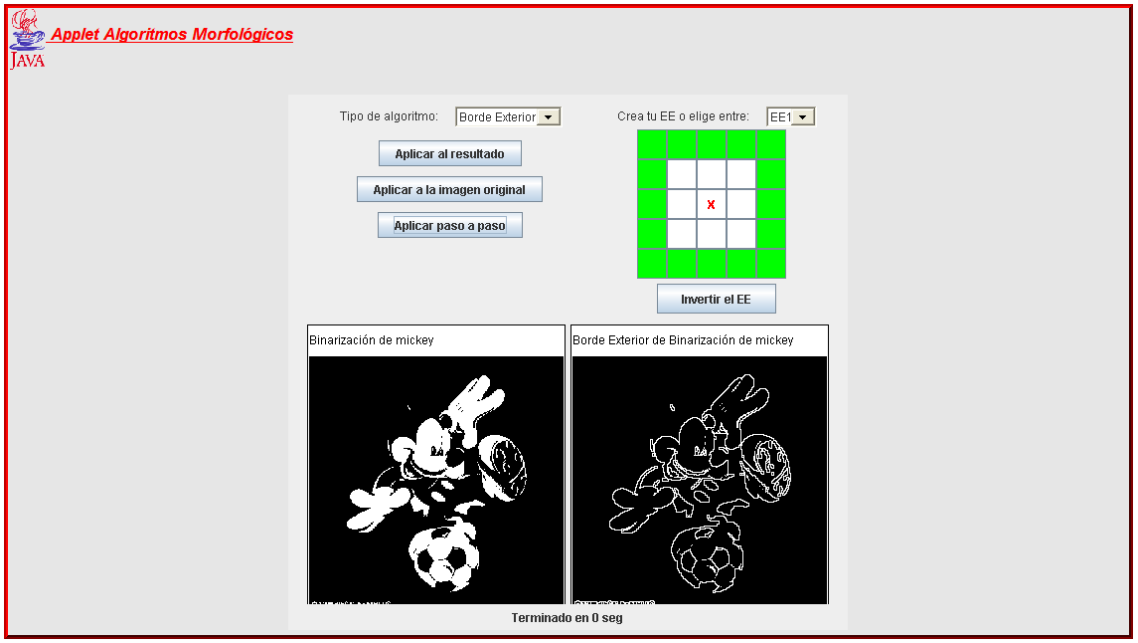


Figura 4.34: Applet Algoritmos morfológicos.

4.2.2.2.4. Applet “Operaciones morfológicas sobre imágenes en escala de grises”

Los fundamentos en los que se basa todo el procedimiento de procesado morfológico en imágenes en escalas de grises son los mismos que en caso de imágenes binarias, eso si, con el correspondiente incremento de complejidad, ya que ahora no estamos manejando valores de intensidad de 0 o 1 (de las imágenes binarias), sino rangos entre 0 y 255 (imágenes a escala de grises).

Por eso, este applet es idéntico al de “Operaciones morfológicas sobre imágenes binarias”, con la salvedad de que en este caso se aplican los operadores morfológicos directamente sobre la imagen en escala de grises, y no sobre la imagen binarizada de ésta.

Las operaciones morfológicas que se pueden realizar son las clásicas (dilatación, erosión, apertura y cierre) y además se incluyen las de s*uavizado*, *gradiente* y *sombrero de copa*.

Nótese que el procesado morfológico en el caso de imágenes en escala de grises es menos evidente (visualmente hablando) que en el caso de imágenes binarias, por eso quizá didácticamente es menos ilustrativo que los applets anteriores.

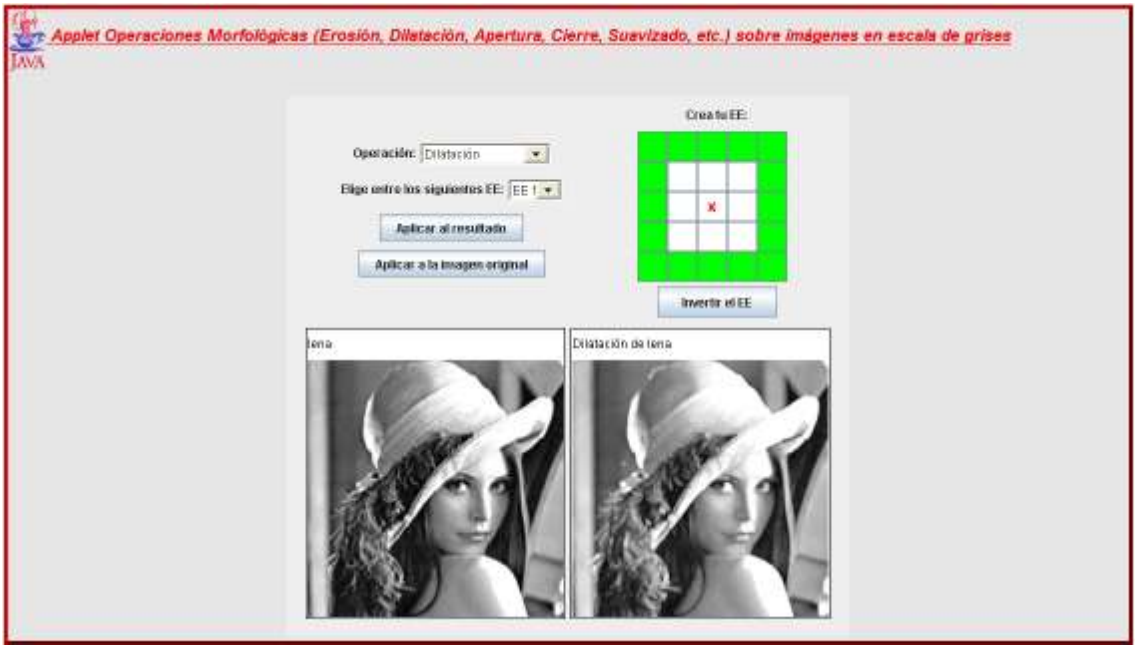


Figura 4.35: Applet Operaciones morfológicas sobre imágenes en escala de grises.

4.2.2.3. APPLETS CURSO ‘RESTAURACIÓN DE IMÁGENES’

Los applets de este curso reúnen varios elementos del curso de procesado básico de imágenes (visualización y ecualización del histograma de la imagen, filtrado de ruido…), pero esta vez también extendidos a imágenes a color. Además se incluyen otras funcionalidades nuevas, como la aplicación de distorsiones geométricas a las imágenes, entre otras.

4.2.2.3.1. Applet "Ecualización del Histograma"

Este applet pretende visualizar de una manera bastante intuitiva los efectos que tiene sobre el contraste de una imagen la igualación de su histograma. Para ello mostramos dos imágenes, la original y la ecualizada. Inmediatamente debajo de cada una de ellas se muestra su histograma.

Podemos observar que la imagen con distribución del histograma más uniforme presenta un mejor contraste.

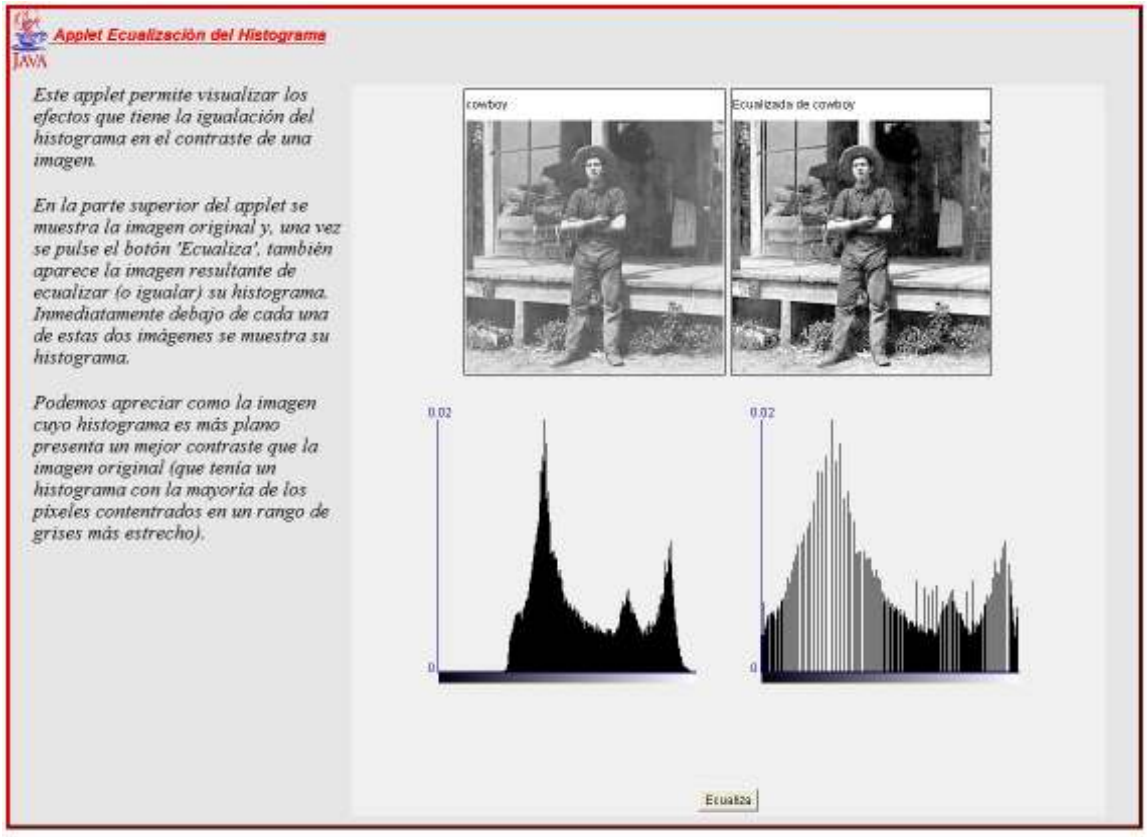


Figura 4.36: Applet Ecualización del histograma.

4.2.2.3.2. Applet "Cancelación de Interferencias Sinusoidales"

Con este applet se intenta ilustrar el efecto sobre la imagen de modificar su Transformada de Fourier.

Esta técnica es muy útil para la cancelación de interferencias sinusoidales, ya que en la imagen original serían difíciles de eliminar, mientras que en su Transformada de Fourier aparecen localizadas en puntos concretos cuya eliminación es más sencilla.

En el panel del applet aparece la imagen original con su TF debajo. Junto a ésta última se muestra una máscara que te permite eliminar puntos de la transformada, apareciendo el resultado en la imagen de la derecha. Finalmente, en la parte superior a la derecha de la imagen original, se muestra la imagen resultado de aplicar la inversa de la transformada de Fourier sobre la TF modificada por la máscara.

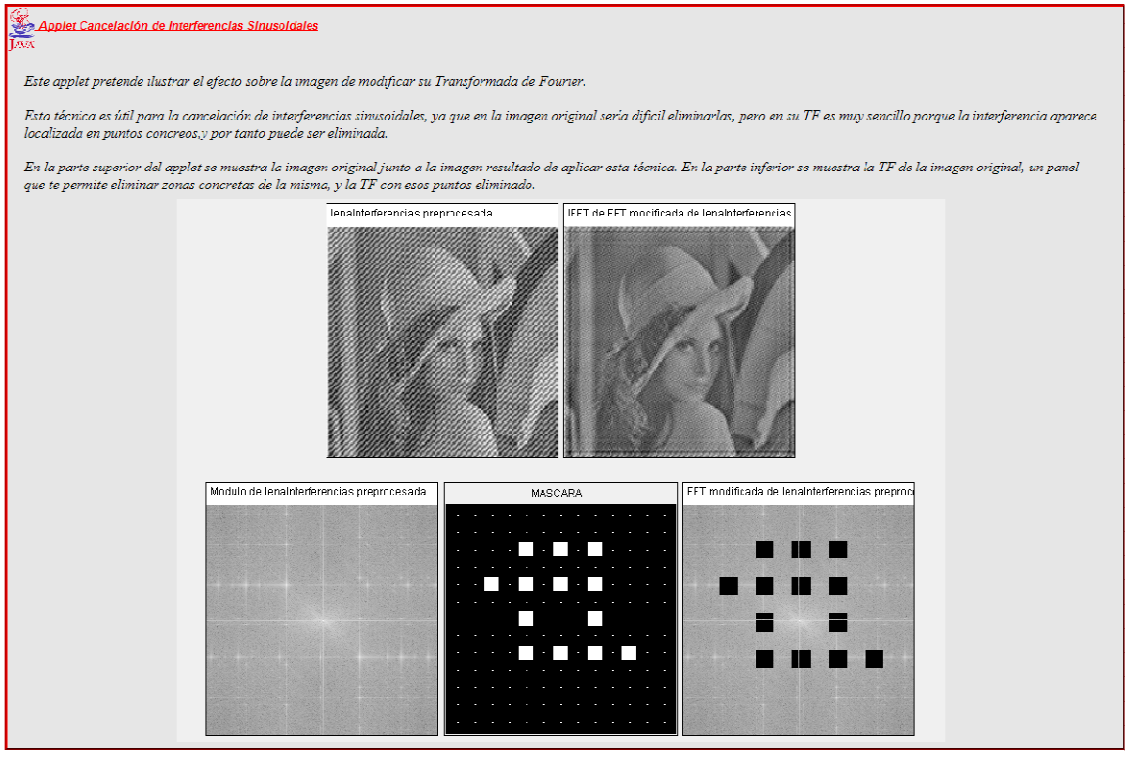


Figura 4.37: Applet Cancelación de Interferencias Sinusoidales.

4.2.2.3.3. Applet "Eliminación de Ruido"

No es lo mismo una imagen con ruido gaussiano que una con ruido impulsivo (efecto sal y pimienta), y por tanto las formas mas eficaces de eliminarlo tampoco son iguales.

Mientras que para una imagen con ruido impulsivo aplicar un filtro de mediana es la solución óptima, la mejor solución para eliminar el ruido gaussiano de una imagen es hacerle un filtrado paso bajo.

Con este applet pretendemos que el usuario pueda experimentar las distintas combinaciones para eliminar ruido jugando con los parámetros que se muestran, y finalmente llegue a conclusiones como la expresada anteriormente.

Además podrá sopesar los pros y los contras de cada técnica de restauración de imágenes ruidosas. Por ejemplo, el filtrado paso bajo elimina el ruido impulsivo de la imagen de una manera más o menos eficaz, pero en contrapartida también se produce un suavizado en los bordes de la imagen.

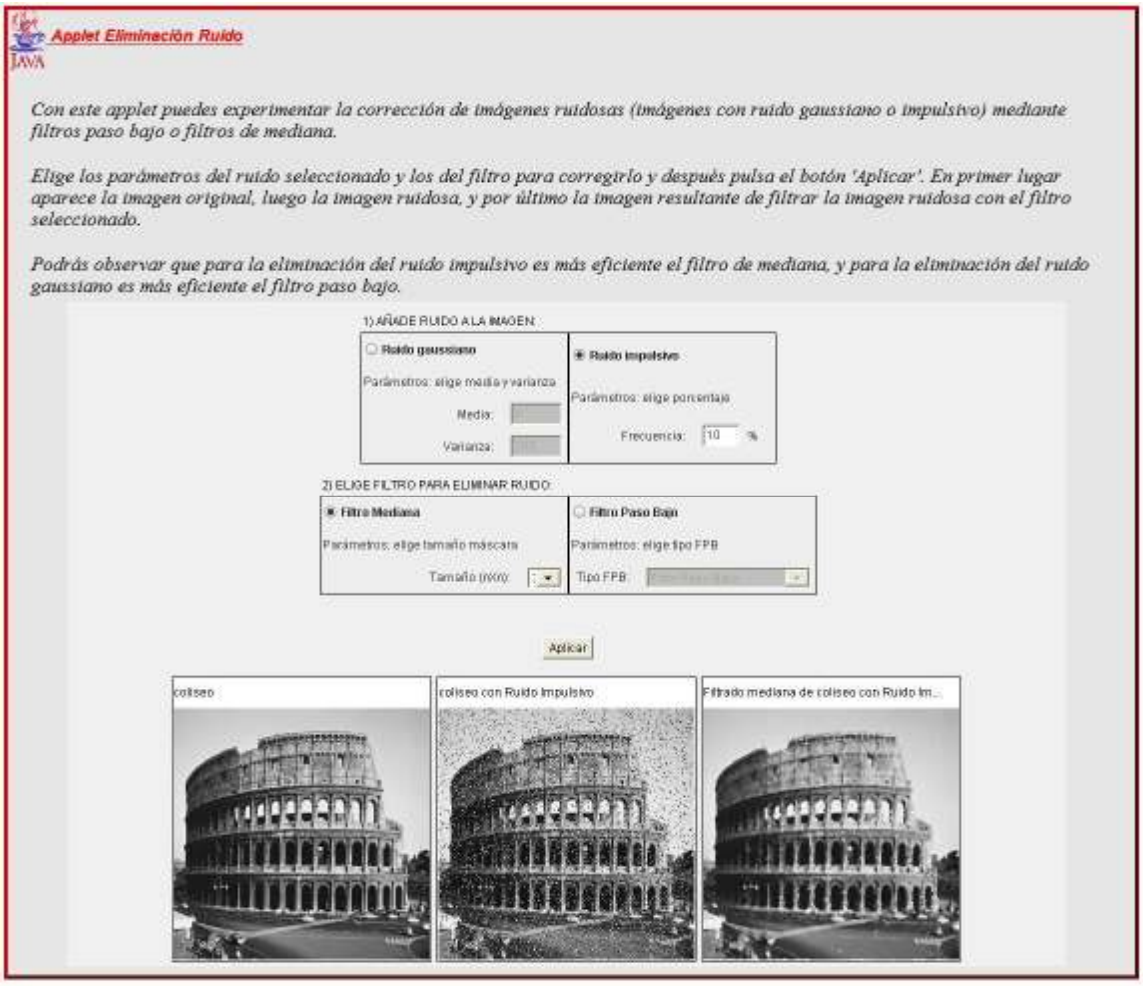


Figura 4.38: Applet Eliminación de ruido.

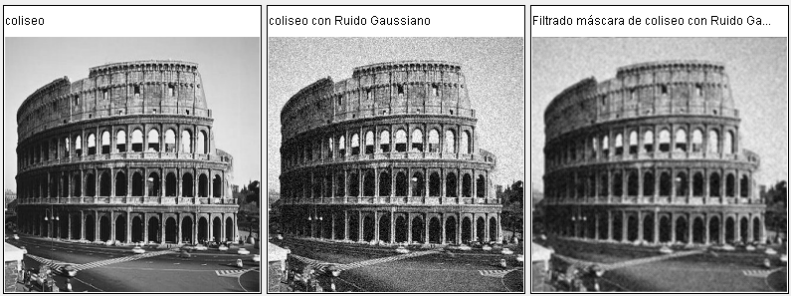


Figura 4.39: Ejemplo técnicas eliminación de ruido: 1)Imagen original. 2)Img 1 con ruido gaussiano. 3) Img 2 tras filtrarla paso bajo.

4.2.2.3.4. Applet "Distorsiones Geométricas"

Se pueden realizar transformaciones sobre las coordenadas de los píxeles de una imagen, modificando con ello las relaciones espaciales entre sus píxeles.

Estas transformaciones pueden ser lineales (obedecen a una función en la que, dados unos valores de entrada, obtenemos los correspondientes valores de salida), o no lineales (por ejemplo las distorsiones pin-cushion o barrel).

Con este applet se puede practicar los efectos de algunas de estas distorsiones geométricas (tanto lineales como no lineales) sobre una imagen. Aparece un combo con varias distorsiones (barrel, pin-cushion, distorsion-horizontal 1/2/3 y distorsion-vertical 1/2/3), cuya forma se muestra a la derecha al pinchar sobre ellas. Finalmente, cuando el botón 'Aplicar' es pulsado, se aplica la distorsión geométrica seleccionada a la imagen original.

Como en el resto de los applets, se puede cargar otra imagen pulsando con el botón derecho del ratón sobre la imagen original mostrada inicialmente, y seleccionando después la opción 'Cargar imagen' del menú desplegable.

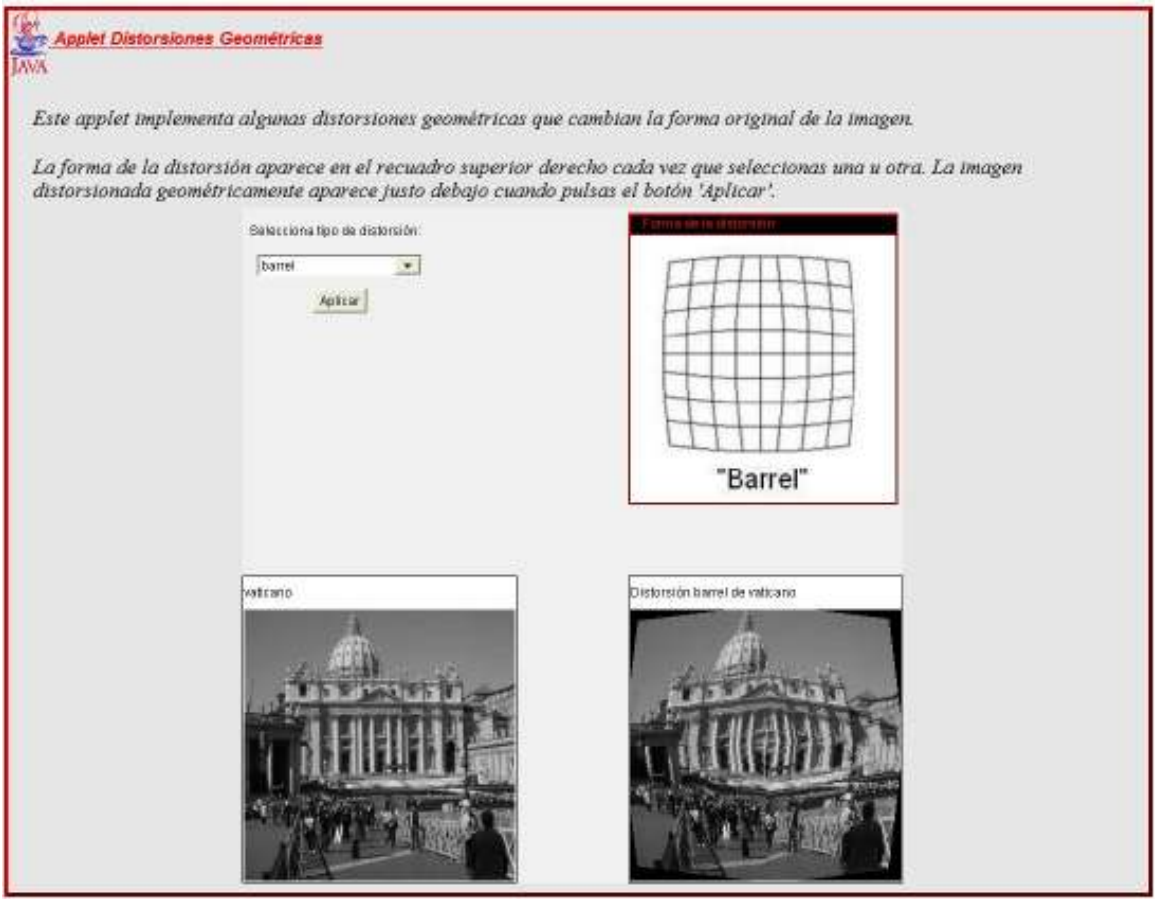


Figura 4.40: Applet Distorsiones Geométricas.

4.2.2.3.5. Applet "Imágenes en color"

Este applet contiene las funcionalidades principales del tratamiento digital de imágenes en color. Para experimentar con ellas, basta con pinchar sobre la imagen con el botón derecho del ratón y seleccionar la opción deseada en el menú desplegable.

Las opciones que incluye son:

- *Muestra histogramas componentes RGB:* muestra los histogramas por separado de las tres componentes de la imagen (rojo, verde y azul).

- *Convierte a escala de grises:* convierte la imagen a color en una imagen a escala de grises.

- *Reescala imagen:* aplica un escalado a la imagen. Para ello se pueden establecer las dimensiones de la imagen resultante o aplicar un factor de escala a la imagen original.

- *Filtra la imagen:* aplica un filtro a la imagen definido por una matriz de coeficientes (pudiendo elegir unos valores predefinidos que implementan un filtro paso bajo, uno paso alto, etc. o bien introduciendo los valores que quiera el usuario).

- *Gira la imagen:* rota la imagen tantos grados como introduzca el usuario

- *Cambia intensidad de una componente (R, G o B*): cambia el valor de la intensidad de la componente de la imagen seleccionada (rojo, verde o azul) para todos los píxeles de la imagen.

- *Transformada de Fourier:* muestra la TF de la imagen (modulo-fase y parte real-imaginaria).

- *Cargar imagen:* muestra las imágenes a color disponibles en el servidor permitiendo cargar cualquiera de ellas.

Todas estas opciones estarán disponibles en el resto de applets del curso que utilicen imágenes a color (pinchando con el botón derecho del ratón sobre la imagen a color).

Figura 4.41: Applet Imágenes a Color.

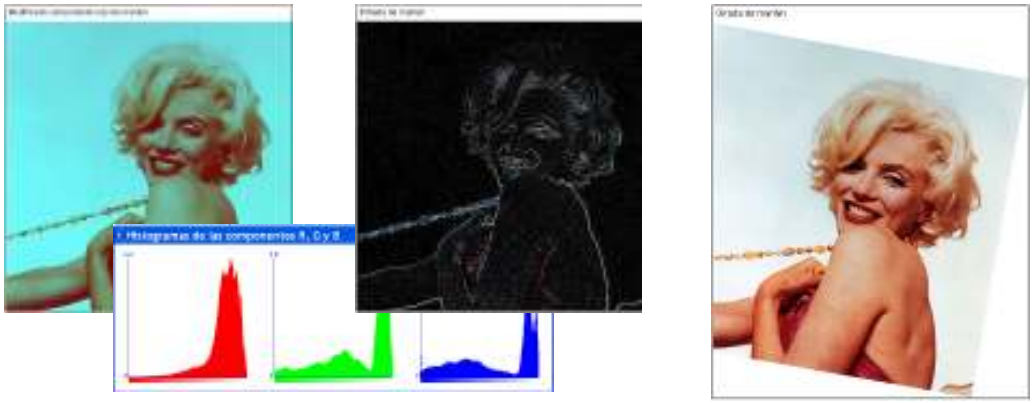


Figura 4.42: Ejemplos operaciones sobre imágenes a color.

4.2.3. APPLETS: FUNCIONAMIENTO A NIVEL TÉCNICO

Cada applet que compone el curso está construido mediante clases java que le dan la funcionalidad que deseamos en cada caso. Algunas de ellas son específicas del applet en cuestión (por ejemplo, aquellas que crean su apariencia), pero hay clases comunes que contienen la funcionalidad básica de la aplicación (por ejemplo la carga de imágenes y las técnicas de procesado básicas).

Para hacernos una idea de la estructuración del proyecto, se puede revisar la [*Figura 4.16: Estructura clases de IMAGine en este proyecto*], donde vemos la actual organización en cursos con los applets que contienen esos cursos, en paquetes con clases comunes para todos ellos y en la agrupación de aquellas que gestionan la base de imágenes (carga, edición, etc.).

Por tanto, en este apartado explicaremos cada uno de los packages del proyecto, que los dividiremos en tres grupos:

- Clases que gestionan la base de imágenes de la aplicación (carga, edición y recuperación de imágenes de ella).

- Clases comunes a todos los applets que contienen la funcionalidad básica de IMAGine (definición de imagen a color, imagen a escala de grises, contenedores de ambas y submenús específicos para cada una de ellas con las opciones básicas de procesado de cada una de ellas).

- Clases específicas de los applets divididas en cursos que se subdividen en cada uno de los applets de la aplicación. En ellos, se analizará la estructura común que implementan todos y explicaremos que clases utilizan a nivel de interfaz, de implementación del applet y de funcionamiento. Las clases más importantes serán explicadas en detalle, mientras que las que conforman la apariencia del applet (caracterizadas por llevar la palabra *Panel* o *Canvas* al final) o las interfaces de constantes serán tratadas más superficialmente al ser menor su relevancia técnica.

4.2.3.1. Package BaseImagenes: Clases que se ocupan de la base de imágenes

Estas clases se encargan de gestionar todo lo relacionado con el acceso, carga, edición y recuperación de imágenes del directorio donde se alojan las imágenes disponibles en los applets de IMAGine, lo que también llamaremos Base de Imágenes del Servidor.

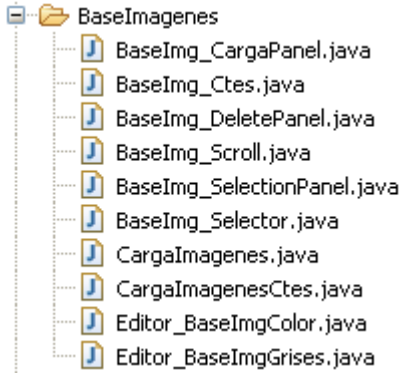


Figura 4.43: Package BaseImagenes.
Clases que se ocupan de la base de imágenes.

Actualmente, hay dos directorios donde se alojan las imágenes que utilizamos. Por un lado está la base de imágenes a color y por otro lado la de las imágenes en escala de grises. Esta separación viene dada porque hay diferencias en el tratamiento de estos dos tipos de imágenes:

Las imágenes en escala de grises, al grabarlas para que puedan ser utilizadas por la aplicación, se editan dando como resultado una imagen en escala de grises, de tamaño cuadrado de dimensiones 256x256. Cualquier variación de estos valores provoca que muchas de las técnicas de procesado de imágenes implementadas en proyectos anteriores para ellas fallen, por lo que hemos considerado conveniente mantener esta funcionalidad.

En cambio, para las imágenes a color (cuya funcionalidad se ha implementado en este proyecto fin de carrera), se ha tenido en cuenta esa limitación que ocurría con la base de imágenes implementada anteriormente, salvándola exitosamente. En la base de imágenes a color se almacenan las imágenes con su tamaño original, adaptando las técnicas de procesado implementadas para ellas teniendo en cuenta ese factor, lo que aporta una mayor flexibilidad a los applets que se crean con este tipo de imágenes.

Por tanto, se hacía necesario también una aplicación distinta para cargar cada uno de estos tipos de imágenes en el formato adecuado al directorio de imágenes. Por eso tenemos dos Editores, aunque la funcionalidad es muy parecida:

-En el caso de imágenes en escala de grises se edita la imagen antes de grabarla en el directorio final (cuadrada, convertida a grises…), mientras que con las imágenes a color se graba idéntica a la original. Se ha modificado la carga de imágenes para que se graben con extensión .jpg y no serializadas como se hacía antes (esto facilita el mantenimiento de la BBII al poder visualizarlas con cualquier aplicación de imágenes). [Imagen: extensión *.jpg*]

-En ambos casos, también se crea una imagen en miniatura para mostrarla en el panel de selección de imágenes. [ImagenMiniatura: extensión *.min.jpg*]

-Se ha eliminado la creación de la FFT de la imagen que se pregrabarla en la base de imágenes debido a que los tiempos actuales de procesamiento no hacen necesario este procedimiento, ya que la generación de la misma en tiempo real es suficientemente rápida.

-Se crea un listado .txt para cada base de imágenes con los nombres de las imágenes contenidas en cada una. Esto nos permite acceder a ellas desde el applet cuando estamos ejecutándolo en el servidor. El formato de este listado ha sido modificado a un archivo de texto plano para facilitar el mantenimiento de la BBII, ya que antes al grabarse serializado, no se podía modificar manualmente.

Pasamos a analizar cada una de las clases que componen este paquete para ver la funcionalidad real de cada una:

- *Editor_BaseImgGrises* y *Editor_BaseImgColor:*

Es la encargada de gestionar la base de imágenes del servidor (Editor_BaseImgGrises gestiona la BBII a escala de grises y Editor_BaseImgColor se encarga de la BBII a color).

Tiene un método main que lanza su ejecución (no es un applet y por lo tanto no hay que ejecutarlo en un navegador). Proporciona la interfaz gráfica que nos permite añadir imágenes desde el disco duro a la base de imágenes (un directorio que luego subiremos al servidor), además de borrar imágenes ya existentes en dicho directorio. En cualquier momento se puede ver el estado de la base de imágenes.

En el caso de carga a la BBII de imágenes a escala de grises, la imagen cargada desde local se editará para que cumpla con el formato definido (convierte las imágenes para que sean en blanco y negro, añade bandas negras a la imagen para que ésta sea cuadrada…). En el caso de carga a la BBII de imágenes a color, no será necesaria esta edición grabándose idéntica a la original.

Al guardar una nueva imagen en el servidor, el programa también guarda la versión en pequeño de la imagen para el panel de selección de imágenes. Por tanto, se graban 2 archivos con distintas extensiones:

- .jpg: Este archivo contiene la imagen original (imágenes para BBII a color) /imagen original editada (imágenes para BBII a escala de grises)
- .min.jpg: Este archivo contiene la miniatura de la imagen original para el panel de selección de imágenes.

Además existe un archivo llamado 'listadoImagenesGrises.txt'/ 'listadoImagenesColor.txt' que guarda la información de todas las imágenes que se almacenan en la BBII para que el ordenador sea capaz de acceder a ellas (este fichero es modificado al guardar y eliminar imágenes).

Figura 4.44: Programas para gestionar la bases de imágenes de IMAGine
(Funcionamiento de la clase Editor_BaseImgGrises.java y Editor_Base_ImgColor.java)

Para cargar una imagen del disco duro del ordenador a la base de imágenes habrá que pulsar primero el botón 'Fichero'. A continuación seleccionamos la imagen a cargar y por último pinchamos sobre el botón 'Añadir imagen' (cuando hayamos seleccionado la imagen del disco duro del ordenador esta se previsualizará en el contenedor de la esquina inferior izquierda, y su nombre aparecerá en el campo 'Nombre de la imagen').

Para hacer la carga desde una URL habría que introducir una URL en el campo que viene destinado a tal fin en el programa y a continuación pulsar el botón 'Cargar'. Finalmente pulsar el botón 'Añadir imagen'.

Para borrar imágenes del servidor hay que pulsar el botón 'Estado actual de la base de imágenes' y a continuación seleccionar en el panel que aparece las imágenes que se desean borrar. Una vez seleccionadas, pulsar el botón 'Eliminar'.

• *BaseImg_Ctes:* Contiene las variables necesarias para el Editor de la BBII (los nombres de los botones que al ser pulsados desencadenan determinados eventos). Además, define una serie de constantes de las clases que gestionan la base, como el ancho/alto de los paneles con las miniaturas de las imágenes, el número máximo de imágenes que se pueden seleccionar en la carga, etc.

• *BaseImg_SelectionPanel:* Panel que contiene las miniaturas de las imágenes cargadas en la base de imágenes ordenadas en filas y columnas en función a la anchura fijada para la ventana y al número de imágenes cargadas en la BBII. En el se puede seleccionar una imagen (en el caso de 'Carga') o varias (en el caso de 'Eliminar').

• *BaseImg_Selector:* clase que sirve para seleccionar imágenes del panel BaseImg_SelectionPanel. Pinta un reborde sobre ellas cuando están pulsadas. Para hacer esto, añade un escuchador de eventos a cada imagen del panel para saber cuando se pulsa con el ratón sobre ellas. Así podemos saber las seleccionadas.

• *BaseImg_Scroll:* Panel con scrolls (barras de desplazamiento) que permiten ver el panel BaseImg_SelectionPanel en una ventana de unas dimensiones concretas aunque el panel con las imágenes tenga un tamaño mayor (para ello utilizaremos las barras de desplazamiento). Permite obtener el listado de imágenes cargadas en el panel (imágenes cargadas por tanto en la BBII) y el listado de las imágenes que se hayan seleccionado en él.

• *BaseImg_DeletePanel.java:* Panel que muestra el panel con scrolls BaseImg_Scroll mas un botón 'Eliminar'. Por tanto, en él aparecen las imágenes almacenadas en la base de imágenes del servidor más un botón para eliminar las que se seleccionen. Cada vez que queramos borrar una imagen tendremos que borrar los dos archivos referentes a la imagen, y también eliminar su nombre del archivo que contiene la lista de las imágenes que se encuentran en la base de imágenes. Este panel aparece al pulsar el botón 'Estado actual de la base de imágenes' del programa Editor de Base de Imágenes. Lo llamaremos panel de 'Eliminar Imagen' de la BBII.

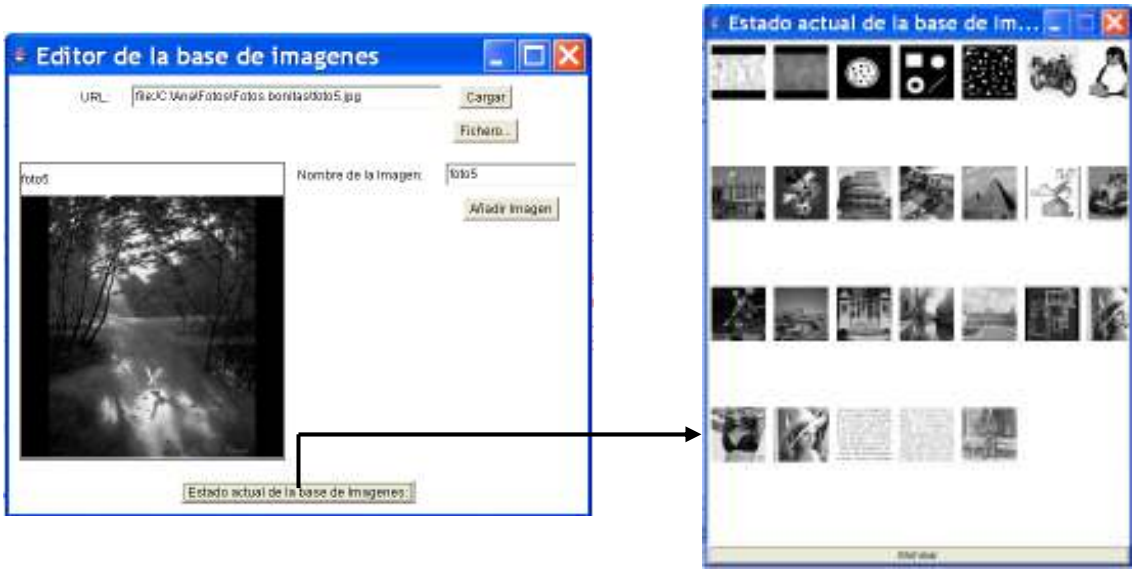


Figura 4.45: Panel 'Eliminar imágenes' de la base de imágenes del servidor. Muestra también el Estado actual de la base de imágenes. (Funcionamiento de la clase BaseImg_DeletePanel.java)

• *BaseImg_CargaPanel.java:* Panel compuesto por el panel con scrolls BaseImg_Scroll mas un botón 'Cargar'. Por tanto, crea el panel donde se podrá elegir cargar en el applet una imagen de las que haya en la base de imágenes del servidor. Se ejecuta al pulsar en la opción 'Cargar imagen' del submenú desplegable que aparece al pinchar con el botón derecho del ratón sobre cualquier imagen de un applet. Denominaremos este panel como panel de 'Cargar Imagen' en el applet.

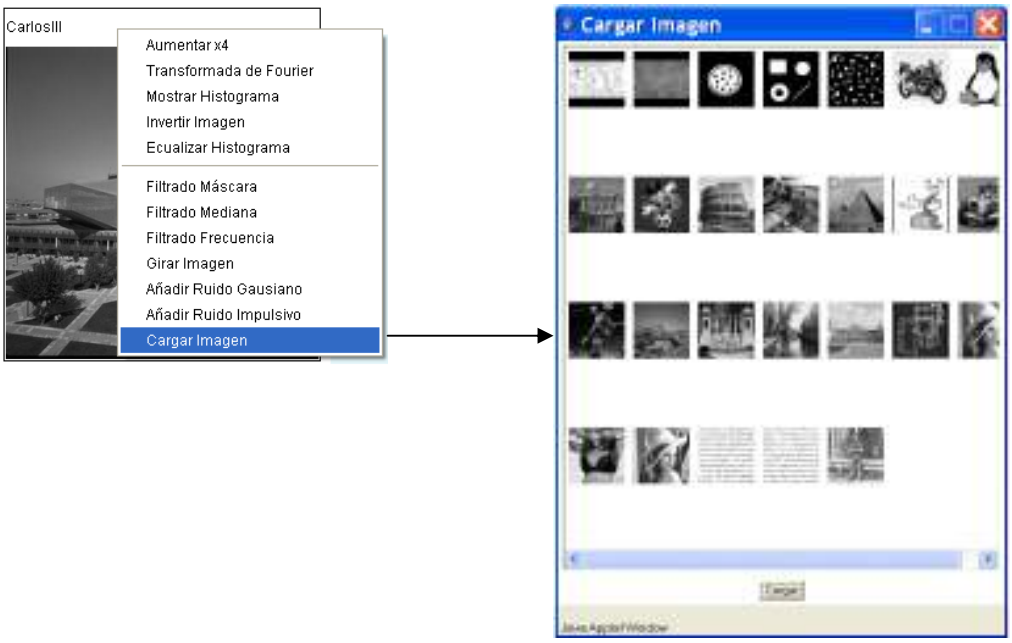


Figura 4.46: Panel 'Cargar imagen' en el applet.
(Funcionamiento de la clase BaseImg_CargaPanel.java)

• *CargaImagenesCtes.java:* Esta clase es de suma importancia ya que contiene todas las constantes de configuración de la grabación/recuperación de imágenes en sí: directorio donde se almacenan las imágenes y desde donde se recuperan cuando ejecutas los applets o el Editor (nótese que el editor siempre va a obtener las imágenes desde un directorio en local, pero los applets se puede ejecutar en local o en el servidor, y por tanto en ese caso las rutas de las imágenes deben ser URLs). Extensión de las imágenes almacenadas y de las miniaturas. También los nombres de los archivos que contienen los listados de las imágenes contenidas en los directorios que constituyen las distintas bases de imágenes. Todos ellos son parámetros imprescindibles para que funcione la aplicación, por tanto es necesario configurarlos correctamente.

• *CargaImagenes.java:* Esta clase contiene toda la funcionalidad real de recuperación de las imágenes almacenadas en la BBII. Las imágenes se recuperan desde la base como objetos java.awt.Image. Se pueden obtener indicando el nombre de la imagen que queremos recuperar o la posición que ocupa en la base de imágenes. Después, dependiendo de si lo que estamos recuperando es una imagen de las de la BBII a color o de la de grises, convertimos la Image a un objeto ImagenColor o Imagen respectivamente, estableciendo atributos como el título de la imagen recuperada.

4.2.3.2. Package comun: Clases que contienen la funcionalidad básica de la aplicación

Estas clases contienen la funcionalidad básica de IMAGine (definición de imagen a color, imagen a escala de grises, contenedores de ambas y submenús específicos para cada una de ellas con las opciones básicas de procesado de cada una de ellas, función FFT, etc.).

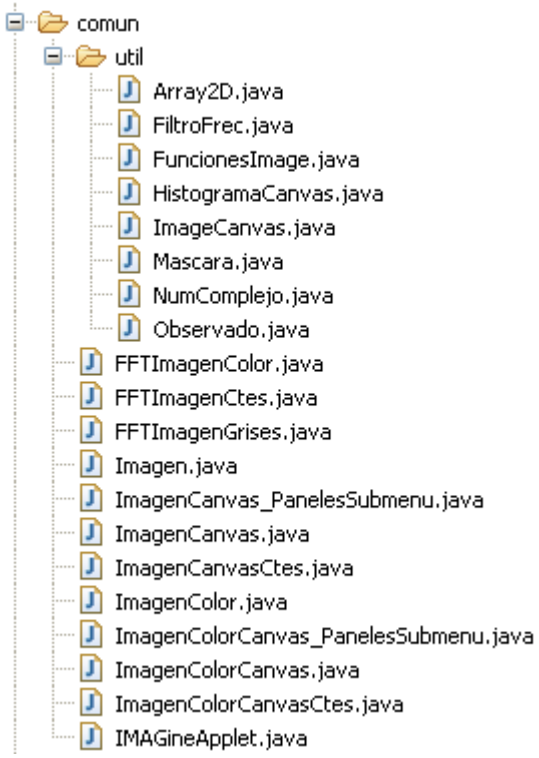


Figura 4.47: Package comun.
Clases que contienen la funcionalidad básica de IMAGine.

• *IMAGineApplet.java:* clase que extiende de Applet y de la que heredaran todos los applets de la aplicación. Contiene una variable que nos permite saber cuando se están obteniendo las imágenes desde el Editor de la base de imágenes o desde los applets (es necesario saberlo para establecer la ruta completa desde la que se obtendrán las imágenes de la BBII, porque se construyen de forma distinta). Tiene los métodos básicos para obtener de forma sencilla una imagen a escala de grises o a color a partir de su nombre o su posición en la BBII.

Clases que implementan la funcionalidad de Imágenes en escala de grises:

• *Imagen.java:* esta clase es una de las más importantes de IMAGine, por eso la explicaremos al detalle.

Contiene la funcionalidad básica (implementa una imagen digital cargada en la BBII de imágenes a escala de grises con ciertas propiedades: almacena los píxeles que componen dicha imagen como arrays bidimensionales de enteros, su título, sus dimensiones y su transformada de Fourier si fuese necesario calcularla, entre otras cosas. Además tiene los métodos para aplicar sobre la imagen todas las técnicas de procesado implementadas en la aplicación para este tipo de imágenes.

- Crea una Imagen (como un array bidimensional de enteros):
  - · Con la primera imagen de la base de imágenes
    `public Imagen()`
  - · Con la imagen cuya posición le pasamos por parámetro
    `public Imagen(int i)`
  - · Como copia de otra imagen.
    `public Imagen(Imagen im)`
  - · Crea una imagen en negro de las dimensiones que indiquemos.
    `public Imagen(int wi, int he)`
  - · A partir de un objeto Image que contenga una imagen:
    `public Imagen(Image image)`
  - · A partir de una URL:
    `public Imagen(URL ucompleta, String tit)`

- Métodos auxiliares para los constructores (Imagen⇔Image):
  - · Carga de imágenes a partir de un objeto Image *(crea un objeto Imagen a partir de un objeto Image)*
    `private void creaImagen(Image image):`
  - · Crea un objeto Image a partir de un objeto Imagen:
    `public Image creaImage()`

- Métodos auxiliares para obtener/poner:
  - · El título de la imagen
    `public String getTitulo()` / `public void setTitulo(String nuevoTitulo)`
  - · Los píxeles que componen la imagen
    `public int[][] getPixels()` / `public void setPixels(int newPixels[][])`
  - · La altura o anchura de la imagen
    `public int getWidth()` / `public void setWidth(int newWidth)`
    `public int getHeight()` / `public void setHeight(int newHeight)`
  - · Obtener el número de iteraciones de un proceso morfológico.
    `public int getIteraciones()`

- Resto de métodos auxiliares para distintas funcionalidades:
  - · Centrar la imagen (la hace cuadrada de tamaño size y añade bandas negras si es necesario)
    `public void centra(int size)`
  - · Reescalar el rango de grises para que coincida con el rango dinámico
    `public void cropscale()`
  - · Adaptar el módulo de la TF para que los ejes aparezcan centrados en la imagen
    `public void fftshift()`

Además, proporciona los métodos para implementar el funcionamiento de todas las opciones del submenú desplegable que aparece al pinchar con el botón derecho del ratón sobre cualquier imagen de un applet.

- Zoom de la imagen
  public Image zoomImage(int x, int y, int ancho, int alto, int factor)

º Obtiene/Pone la TF de una imagen *(funcionalidad implementada en la clase FFTImagen)*
  public FFTImagen getTf() / public void setTf(FFTImagen newTf)

- Invertir los tonos de la imagen
  public void invierte()
- Ecualizar el histograma de la imagen
  public void ecualizaHistograma()
- Calcular el histograma de la imagen
  public int [] histograma()
- Girar la imagen un cierto ángulo
  public void gira(float radianes)
  public void gira(float radianes,boolean mas_cercano)
- Filtrado con máscara
  public void filtra(Mascara m)
  → public void margenCorta()
  → public void margenDinamico()
  → public void cortaCero()
- Filtrado de mediana
  public void filtraMediana(int tam)
- Aplicar ruido gaussiano
  public void ruidoGaussiano(float media, float varianza)
- Aplicar ruido impulsivo
  public void ruidoImpulsivo(float porcentaje)

También contiene los métodos principales para la ejecución de todos los applets del curso (en la mayoría de los casos, estos a su vez llaman a las funciones específicas de la clase donde están implementados los algoritmos en los que se fundamenta su funcionamiento).

-CURSO PROCESADO BÁSICO DE IMÁGENES: *(funcionalidad distinta a las técnicas del submenú implementadas en FFTImagen, FiltroFrec, FiltroMedianaAlgoritmo, etc).*

· Aplicar una función de transferencia determinada a la imagen
  public void transferencia(FuncionTransferencia func)

-CURSO PROCESADO MORFOLÓGICO: *(funcionalidad implementada en las clases ProcMorf y EE)*

º Realiza una operación morfológica puntual a una imagen binaria:
  public void operaMorf(EE e, String operacion)

º Realiza la operación hit or miss:
  public void operaHitMiss(EE e)

º Aplica un algoritmo morfológico:
  public void bordes(EE e, String borde)

º Aplica un algoritmo morfológico:
  public void bordes(EE e, String borde)

º Realiza una operación morfológica puntual a una imagen en escala de grises:
  public void operaMorfGrises

-CURSO RESTAURACIÓN DE IMÁGENES: *(funcionalidad implementada en la clase RestauracionImg)*

º Aplica una distorsión geométrica a una imagen a escala de grises:

public void distorsionGeometrica(String tipoDistorsion)

º Multiplica píxel a píxel dos objetos Imagen:

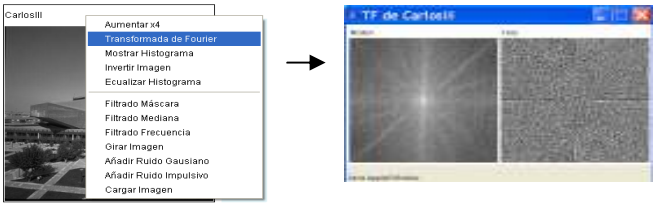
public void multiplicaDosImagenes (Imagen img1, Imagen img2)

• *ImagenCanvasCtes.java:* define todas las opciones del menú emergente de imágenes a escalas de grises (ya que al ser pulsadas desencadenarán eventos para que se realice el procesado que nombran), y los nombres de los botones de los paneles emergentes (ya que al ser pulsados también desencadenarán eventos como aplicar el procesado a la imagen de acuerdo a los parámetros introducidos en el panel).

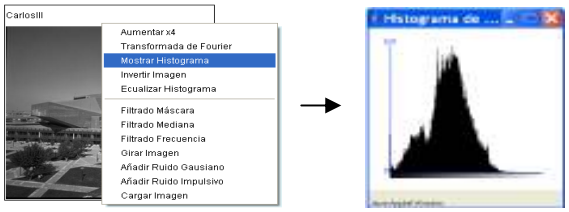
• *ImagenCanvas.java:* contenedor para poder pintar un objeto Imagen en un panel y mostrarlo por pantalla. Se puede interpretar como el *marco de cada una de las imágenes* de la BBII de imágenes a escala de grises. De él podemos obtener datos como la dimensión, la imagen a escala de grises cargada en ese momento, actualizar la imagen contenida en él, etc. También se utiliza para poder implementar el *menú desplegable de imágenes a escala de grises* que nos servirá para realizar las diferentes opciones del tratamiento digital de imágenes que ofrece el BásicoApplet.java. Invoca las clases correspondientes para el funcionamiento de las opciones del menú desplegable y gestiona los paneles que aparecen en alguna de esas opciones al seleccionarlas (abriendo ventanas que muestran los paneles y cerrando las ya existentes).

• *ImagenCanvas_PanelesSubmenu.java:* implementa todos los paneles emergentes que pueden aparecer al pulsar cualquiera de las opciones del menú desplegable de imágenes a escala de grises. También contiene todos los métodos para obtener los valores introducidos en cada uno de esos paneles.

- Panel de la opción 'Transformada de Fourier' del submenú

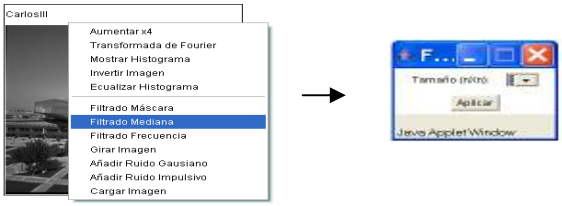


- Panel de la opción 'Mostrar Histograma' del submenú



- Panel de la opción 'Filtrado Máscara' del submenú

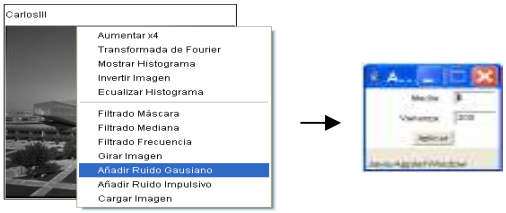
- Panel de la opción 'Filtrado Mediana' del submenú



- Panel de la opción 'Filtrado Frecuencia' del submenú

- Panel de la opción 'Girar Imagen' del submenú

- Panel de la opción 'Añadir Ruido Gaussiano' del submenú



- Panel de la opción 'Añadir Ruido Impulsivo' del submenú

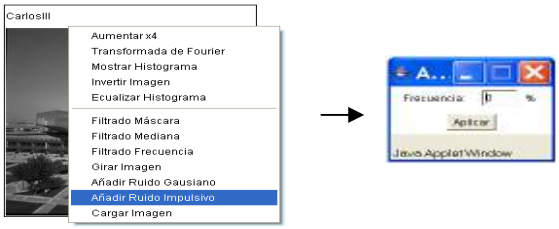


Figura 4.48: Paneles de la clase ImagenCanvas_PanelesSubmenu. Son los paneles emergentes del submenú de imagen grises.

Clases que implementan la funcionalidad de Imágenes a color:

- *ImagenColor.java:* esta clase es otra de las más importantes de IMAGine, y seguramente lo sea más en las sucesivas ampliaciones de IMAGine (es mas interesante y útil estudiar técnicas de procesado de imágenes a color que de imágenes en escala de grises), por lo que procederemos a explicarla en detalle.

Contiene la funcionalidad básica de imágenes a color (implementa una imagen digital cargada en la BBII de imágenes a color con ciertas propiedades: almacena los píxeles que componen dicha imagen como arrays unidimensionales/bidimensionales de enteros, su título, sus dimensiones... Además, tiene los métodos para aplicar sobre la imagen todas las técnicas de procesado implementadas en la aplicación para este tipo de imágenes (filtrado, cambio de la intensidad de alguno de sus componentes R, G o B, rotación, histograma, etc.).

Actualmente, solo se han implementado las técnicas de procesado básicas para imágenes a color, pero quedan abiertas muchas más posibilidades.

Vamos a ver en detalle como está construida internamente esta clase:

- Crea una ImagenColor como un array de enteros unidimensional/bidimensional.
  · Con la primera imagen de la base de imágenes a color
    `public ImagenColor (String nombreImage)`
  · Con la imagen cuya posición le pasamos por parámetro
    `public ImagenColor (int numImagen)`
  · A partir de un objeto java.awt.Image (este es el que realmente tiene la funcionalidad de crear una imagenColor).
    `public ImagenColor (Image image)`

- Método auxiliar para los constructores (Inicializa todos los atributos de la clase a partir del objeto Image creado en los constructores (anchura, altura, array pixels unidimensional y bidimensional y objeto BufferedImage) utilizando métodos estáticos de la clase FuncionesImage.java
  `private void inicializaAtributos(Image image)`

- Métodos auxiliares para obtener los atributos de la clase (no existen los set, excepto del título, porque el resto de los atributos se inicializan a partir de la Image pasada en el constructor, con lo cual para variarlos se tendría que pasar la Image nueva en el constructor y así se inicializarían a los valores adecuados):
  · El objeto Image que contiene la imagen en sí creada en la clase ImagenColor.
    `public Image getImage()`
  · El objeto BufferedImage que nos sirve para el procesado.
    `public BufferedImage getBufferedImagen ()`
  · La altura o anchura de la imagen
    `public int getWidth()`
    `public int getHeight()`
  · El título de la imagen
    `public String getTitulo()` / `public void setTitulo(String titulo)`

- Métodos estáticos para manipular los píxels de una imagen a color. El valor del píxel en una posición del array de enteros que la representa, toma el valor de un entero que codifica los tres colores primarios (rojo, verde, azul) y el componente alpha (el nivel de transparencia). Cada componente se codifica en 8 bits (azul 0-7, verde 8-15, rojo 16-23, alpha 24-31). Por tanto el valor de un pixel será un entero que llamaremos rgb. Trabajar con esto así se hace complicado, por lo que definimos estos métodos estáticos para facilitar el manipulado de los pixels a color.

· Convierte un entero rgb en un array de enteros donde la primera posicion es el valor de la componente R, el segundo el de la G, el tercero el de la B y el cuarto el de la componente Alpha y el método contrario

```
public static int [] getComponentesDeUnPixel (int rgb)
public static int getRGBdeUnPixel (int [] componentesPixel)
```

· Lo mismo que los métodos anteriores pero sustituyendo el array int [] componentesPixel por cuatro enteros que se pasan como parámetros al método.

```
public static int getRGBdeUnPixel (int rojo, int verde, int azul, int alpha)
```

· Obtiene el valor de una componente a partir del entero rgb con el valor de ese pixel:

```
public static int compRojoDeUnPixel (int rgb)
public static int compVerdeDeUnPixel (int rgb)
public static int compAzulDeUnPixel (int rgb)
public static int compAlphaDeUnPixel (int rgb)
```

· Obtiene el valor del entero rgb del píxel al que se le modifica el valor de una de sus componentes:

```
public static int cambiaCompRojoDeUnPixel (int rgb, int newRojo)
public static int cambiaCompVerdeDeUnPixel (int rgb, int newVerde)
public static int cambiaCompAzulDeUnPixel (int rgb, int newAzul)
public static int cambiaCompAlphaDeUnPixel (int rgb, int newAlpha)
```

Además, proporciona los métodos para implementar el funcionamiento de todas las opciones del submenú desplegable que aparece al pinchar con el botón derecho del ratón sobre cualquier imagen a color de un applet.

- Convierte la imagen a color en una imagen con otro espacio de color, por ejemplo a escala de grises

```
public void convierteAotroEspacioDeColor (int espacioDeColor)
```

- Filtra la imagen con los valores de la máscara pasada por parámetros

```
public void filtra (int anchoMatriz, int altoMatriz, float [] matriz)
```

- Reescala el tamaño de la imagen a partir de un factor de escala, o poniendo las dimensiones de la imagen resultante

```
public void reescala (double factorEscala)
public void reescala (int newAncho, int newAlto)
```

- Gira la imagen un cierto ángulo

```
public void gira (double radianes, int tipoInterpolacion)
```

- Calcula los histogramas de cada una de las componentes de la imagen.

```
public int[] histogramaRojo()
public int[] histogramaVerde()
public int[] histogramaAzul()
```

- Modifica la intensidad de todos los píxeles de uno de los componentes de la imagen (“rojo”, “verde” o “azul” a un valor entre 0 y 255).

```
public void modificaIntensidadComponente (String componente, int intensidad)
```

- Obtiene la Transformada de Fourier o la Inversa de la TF de la imagen

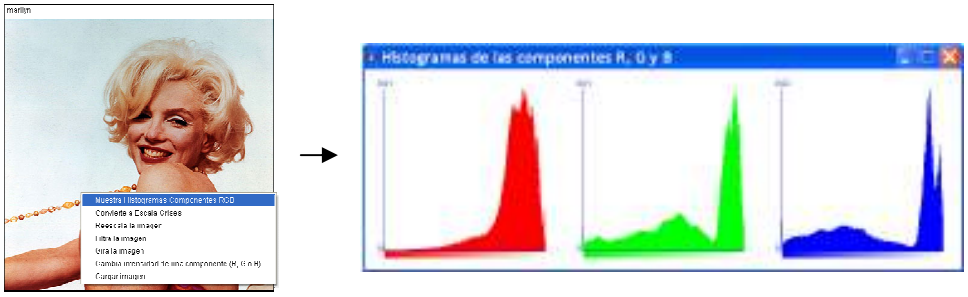
```
public FFTImagenColor getTF () / public FFTImagenColor getInverseTF ()
```

• *ImagenColorCanvasCtes.java:* define todas las opciones del menú emergente de imágenes a color (ya que al ser pulsadas desencadenarán eventos para que se realice el procesado que nombran), y los nombres de los botones de los paneles emergentes (ya que al ser pulsados también desencadenarán eventos como aplicar el procesado a la imagen de acuerdo a los parámetros introducidos en el panel).

• *ImagenColorCanvas.java:* contenedor para poder pintar un objeto ImagenColor en un panel y mostrarlo por pantalla. Se puede interpretar como el *marco de cada una de las imágenes* de la BBII de imágenes a color. De él podemos obtener datos como la dimensión (en este caso el contenedor se ajusta a las dimensiones de la ImagenColor que carguemos en él), la imagen a color cargada en ese momento, actualizar la imagen contenida en él, etc. De manera paralela a lo que ocurría con el contenedor de imágenes en grises, para imágenes a color también tenemos un *menú desplegable de imágenes a color* que nos servirá para realizar las diferentes opciones del tratamiento digital de imágenes a color que ofrece el BásicoColorApplet.java. Invoca las clases correspondientes para el funcionamiento de las opciones del menú desplegable y gestiona los paneles que aparecen en alguna de esas opciones al seleccionarlas (abriendo ventanas que muestran los paneles y cerrando las ya existentes).

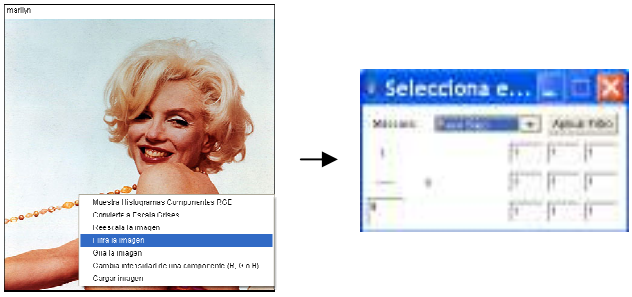
• *ImagenColorCanvas_PanelesSubmenu.java:* implementa todos los paneles emergentes que pueden aparecer al pulsar cualquiera de las opciones del menú desplegable de imágenes a color. También contiene todos los métodos para obtener los valores introducidos en cada uno de esos paneles, haciendo una gestión más transparente de la funcionalidad de la aplicación procediendo de esta forma.

- Panel de la opción 'Mostrar Histogramas Componentes RGB' del submenú imágenes color.



- Panel de la opción 'Reescala la imagen' del submenú imágenes color.

- Panel de la opción 'Filtra la Imagen' del submenú imágenes color.



- Panel de la opción 'Gira la Imagen' del submenú imágenes color.

- Panel de la opción 'Cambia intensidad de una componente (R, G o B)' del submenú imágenes color.

- Panel de la opción 'Tranformada de Fourier' del submenú imágenes color.

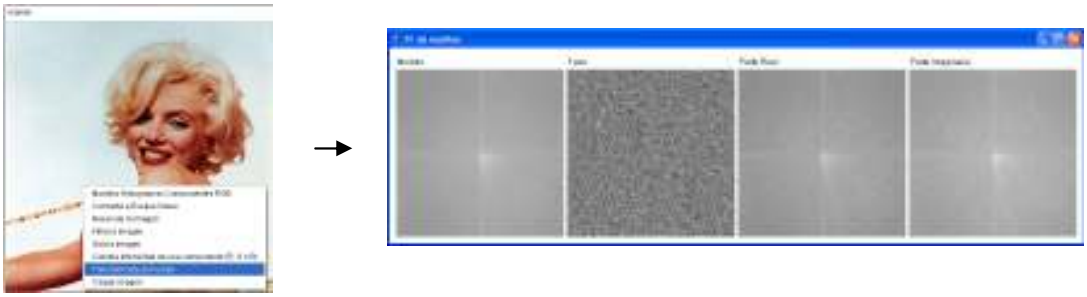


Figura 4.49: Paneles de la clase ImagenColorCanvas_PanelesSubmenu.
Son los paneles emergentes del submenú de imagen color.

Clases que implementan la funcionalidad de la Transformada de Fourier:

- *FFTImagenCtes.java:* constantes necesarias para el funcionamiento de las FFTs de las imágenes (tamaño de la imagen reescalada del módulo de la FFT, etc.).

- *FFTImagenGrises.java:* su función principal es la de calcular la Transformada de Fourier de una imagen de la BBII de escala de grises, así como la transformada inversa. También calcula el módulo y la fase de la transformada, entre otras cosas. Los applets 'Importancia componentes baja frecuencia' y 'Contribución componentes frecuenciales en la DFT' utilizan esta clase simplemente para realizar transformadas de Fourier de imágenes o filtros. Por ejemplo, en el caso del primer applet nombrado, se realiza la TF del Filtro y la TF de la imagen a filtrar y se multiplican ambos. Con la imagen producto hacemos la TF inversa y obtenemos la imagen resultante de haber filtrado la imagen con el filtro seleccionado.

Para realizar los distintos experimentos que se utilizarán en los applets 'Importancia de la fase' 'Experimento del Oppenheim' y 'Promedio de módulos y fases' es necesario también incluir algunos métodos más, como el de poner fase cero a la transformada, el de poner módulo unidad a la transformada, promediar módulo y/o fase de varias transformadas, etc.

- *FFTImagenColor.java:* su función principal es la de calcular la Transformada de Fourier de una imagen de la BBII de color, así como la transformada inversa. También calcula el módulo y la fase de la transformada, la parte real y la imaginaria. Por ejemplo, el applet 'Cancelación de Interferencias Sinusoidales' utiliza esta clase para realizar la FFT y la IFFT de imágenes, para poder cancelar en el dominio frecuencial interferencias sinusoidales.

Clases auxiliares (Package comun.util):

- *FuncionesImage.java:* esta clase contiene los métodos estáticos más importantes necesarios para tratar con objetos de tipo java.awt.Image. A partir de ellos obtenemos los elementos necesarios para aplicar técnicas de procesado sobre las imágenes (podemos obtener los píxeles de una imagen como un array de enteros, crear un objeto BufferedImage necesario para aplicar algunas técnicas de procesado o grabar la imagen en un directorio, etc.). También tenemos los métodos contrarios que nos permiten obtener de nuevo a partir de estos elementos una Image.

- Métodos que obtienen como resultado un objeto java.awt.Image:
  - · Crea un objeto Image a partir de un BufferedImage:
    ```
    public static Image crealmage (BufferedImage bufferImg)
    ```
  - · Crea un objeto Image a partir de un array de pixels bidimensional que representan a la imagen:
    ```
    public static Image crealmage (int [][] pixels)
    ```
  - · Crea un objeto Image a partir de un array de pixels unidimensional que representan a la imagen:
    ```
    public static Image crealmage (int [] pixels)
    ```
  - · Crea un objeto Image a partir de un objeto Component cualquiera:
    ```
    public static Image crealmage(Component component)
    ```

- Métodos para obtener objetos útiles para las técnicas de procesado a partir de un objeto java.awt.Image:

  · Crea un objeto BufferedImage a partir de un objeto Image: este objeto se utiliza para muchas técnicas de procesado y para poder almacenar la imagen en un directorio sin tenerla que serializar (así se graba como .jpg)

  ```
  public static BufferedImage creaBufferedImage(Image image)
  ```

- Métodos para obtener y manejar los pixels de la imagen (como array de enteros unidimensional/bidimensional):

  · Convierte una java.awt.Image en un array de enteros unidimensional con los pixels que la representan:

  ```
  private int[] getPixels (Image image)
  ```

  · Convierte un array de enteros unidimensional en otro bidimensional para poder localizar con mayor facilidad las coordenadas de un píxel mediante su posición en el array (la primera dimensión del array corresponde a la posición en el eje de las 'y' y la segunda dimensión a su posición en el eje de las 'x')

  ```
  private int [][] getArrayPixelsBidimensional (int [] newpixels)
  ```

  · Convierte un array de enteros bidimensional en otro unidimensional (para poder crear a partir de él el objeto Image)

  ```
  private int[] getArrayPixelsUnidimensional (int [][] newpix)
  ```

• *ImageCanvas.java:* contenedor de cualquier objeto Image que sirve para poderlo pintar en pantalla. Se utiliza para mostrar el módulo de las transformadas de fourier, paneles específicos con dibujos sencillos (por ejemplo una cuadricula donde se pintan puntos), o incluso el zoom realizado sobre una imagen.

• *HistogramaCanvas.java:* calcula, crea y muestra el histograma de la imagen. En las imágenes del tipo de la BBII a escala de grises, calculará y mostrará el histograma de los niveles de gris. Para las imágenes de la BBII a color, calculará y creará mostrando por pantalla el histograma de cada una de las componentes de la imagen (R, G y B).

• *Mascara.java:* implementa las máscaras tipo que utilizaremos para el filtrado basado en máscaras de las imágenes (contiene tanto las máscaras que utilizamos para el filtrado de imágenes en escala de grises como las que empleamos en el filtrado de imágenes a color).

• *FiltroFrec.java:* representa un filtro que utilizamos para el filtrado frecuencial de las imágenes a escala de grises.

• *Array2D.java:* estructura de datos que representa un array bidimensional de números complejos. Es necesario para el cálculo de la FFT de imágenes a color.

• *NumComplejo.java:* representa un número complejo. Es necesario para el calculo de la FFT de imágenes a color.

• *Observado.java:* es un escuchador de eventos del applet que nos permitirá saber si se ha producido alguna acción sobre él (desencadenando acciones concretas que definiremos en los métodos update de los applets a los que se haya añadido este escuchador).

4.2.3.3. Package de los applets de los cursos

Estas clases contienen la implementación propiamente dicha de todos los applets de los diferentes cursos. Hay un package por cada curso y distintas subcarpetas para cada applet del mismo. En cada curso hay alguna clase común donde se implementa la funcionalidad específica de las técnicas de procesado de ese curso, como pueden ser determinados algoritmos (algunas veces, las técnicas utilizadas son las mismas que las del menú emergente, por lo que no es necesario implementarlas en las clases comunes del curso de nuevo).

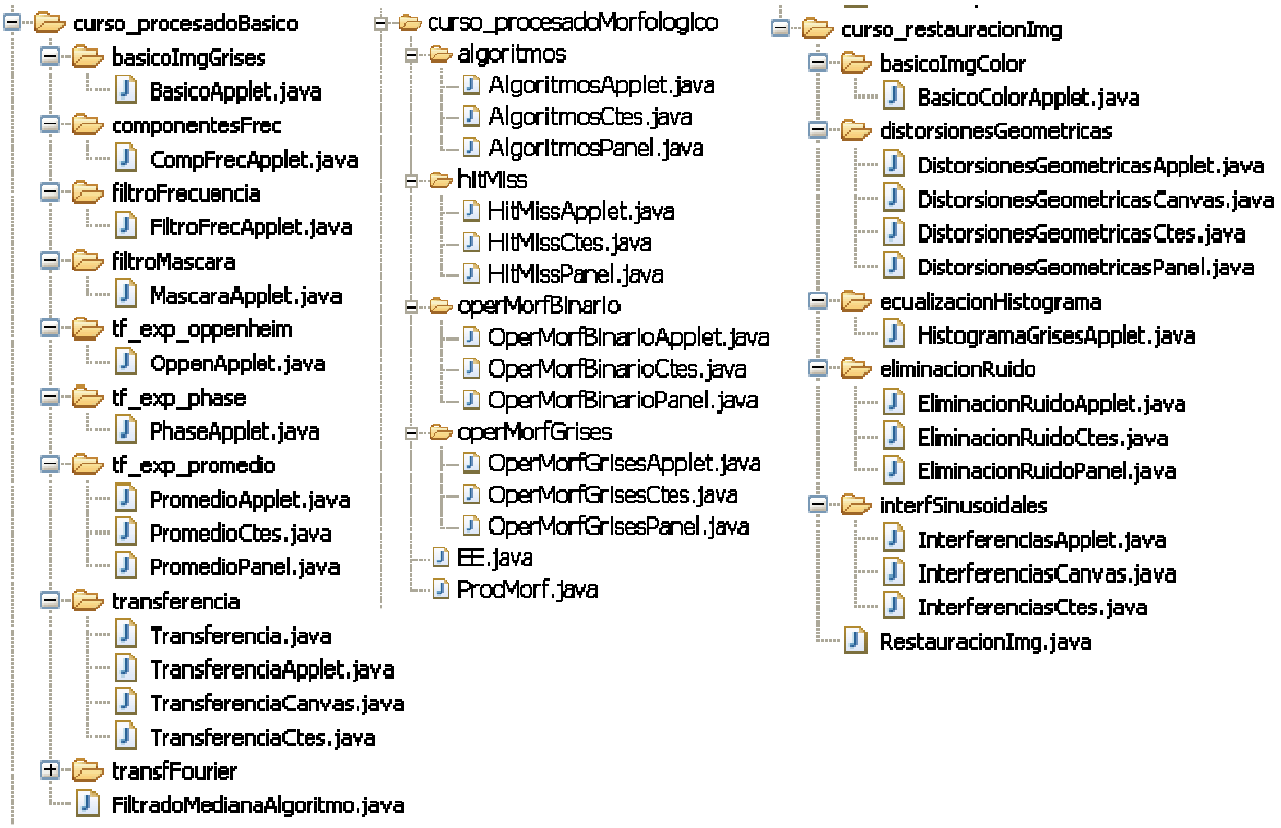


Figura 4.50: Packages de los applets de los cursos.
Clases que contienen la implementación de los applets de IMAGine.

Todos los applets siguen una estructura (a veces se suprime algún elemento por no ser necesario dada la sencillez del applet).

- Interfaz cuyo nombre acaba con el sufijo *Ctes*: contiene las constantes necesarias para el funcionamiento del applet, que suelen corresponder con los nombres de los botones de los paneles que al ser pulsados desencadenan eventos (filtrado de la imagen, aplicar la técnica de procesado seleccionada, etc).

- Clase cuyo nombre acaba con el sufijo *Panel*: implementa el panel que contiene el applet con todos los elementos para definir las técnicas de procesado a aplicar sobre la imagen (combos con distintos tipos de filtros, o checkbox para seleccionar alguna opción, o incluso campos de texto para introducir valores como la media del ruido gaussiano a aplicar…). Esta clase también debe incluir los métodos para obtener los valores introducidos en las casillas del panel.

- Clase cuyo nombre acaba con el sufijo *Canvas*: un applet que tenga una clase con este sufijo es porque en el ha sido necesario crear un contenedor de un objeto Image que contenga algún tipo de dibujo (como por ejemplo dibujo de una malla para situar deltas, o la imagen del resultado de aplicar un tipo de distorsión geométrica, o el módulo de una FFT).

- Clase cuyo nombre acaba con el sufijo *Applet*: es la clase que hereda de IMAGineApplet (que hereda a su vez de java.applet.Applet) y que contiene la implementación del applet en sí controlando de esta forma su ejecución. Hemos definido una estructura interna para este tipo de clases para unificar el código de todos los applets del curso.

En el siguiente ejemplo mostramos un applet básico con un panel y dos contenedores. La estructura interna es idéntica en todos los applets, lo que puede cambiar respecto a otros applets es el método *crearElementosGraficos* (el applet puede tener distinta apariencia) y el método *run* (ahí va la ejecución en sí del applet, aplicando las técnicas de procesado sobre las imágenes). Esta estructura puede variar ligeramente si la técnica de procesado no se aplica al pulsar el botón 'Aplicar' sino al modificar un panel, por ej.

```
public class EjemploApplet extends IMAGineApplet
implements EjemploCtes, Observer,ActionListener,Runnable
{
	/** ************************************************************************
	 *                      Definición de atributos
	 ************************************************************************ */
	/**
	 * Contiene la imagen cargada en el contenedor
	 */
	private Imagen imagenOriginal, imagenModificada;
	/**
	 * Contenedor de la imagen que tiene el applet
	 */
	private ImagenCanvas cimOriginal, cimModificada;
	/**
	 * Panel donde el usuario elige el tipo de procesado a aplicar a la imagenOriginal.
	 */
	private EjemploPanel panelParametros;
	/**
	 * Hilo de ejecución del applet
	 */
	Thread th=null;

	/** ************************************************************************
	 *    Cuerpo de la clase con métodos para la ejecución del applet
	 ************************************************************************ */
	/**  Comienza el applet  */
	public void start()
	{
		//Creamos la apariencia gráfica del applet
		crearElementosGraficos();
	}

	/** Se para el applet */
	public void stop()
	{
		stopThread();
		removeAll();
		cimOriginal=cimModificada =null; panelParametros=null;
		System.gc();
	}
```

```
/** Comienza el hilo de ejecución del applet */
private void startThread()
{
	if(th==null)
	{
		th=new Thread(this);
		th.start();
	}
}

/**  Se para el hilo de ejecución del applet */
private void stopThread()
{
	if(th!=null)
	{
		th.stop();
		th=null;
	}
}

/**
 * Vuelve a activar el hilo de ejecución del applet (para recargar
 * los contenedores con la imagen cargada)
 */
private void filtra()
{
	stopThread();
	startThread();
}

/**
 * Se actualiza el filtrado de la imagen (cuando le damos a cargar otra imagen)
 */
public void update(Observable observable, Object o)
{
	//Se elimina la imagen que pudiese haber en los contenedores
	//con los resultados del procesado de la imagen original porque
	//se ha cargado una nueva imagen y todavía no se ha aplicado ningun procesado

	cimModificada.actualiza(null);
}

/**
 * Contiene el funcionamiento que debe tener el applet cuando se pulsa el botón Aplicar
 * Calcula el procesado de las imágenes y las muestra en los contenedores correspondientes.
 */
public void run()
{
	showStatus("Calculando imagen modificada ");
	//Obtenemos las opciones seleccionadas en el panel para saber que
	//parámetros pasamos  a la técnica de procesado
	String optionSelect=panelParametros.getOptionSelect ();

	//Siempre hacemos las operaciones sobre una copia de la imagen original
	//para dejar esta intacta.
	imagenModificada = new Imagen (cimOriginal.imagen);

	//Aplicamos la tecnica de procesado sobre la imagenModificada estableciendo los
	//parametros seleccionados por el usuario en la tecnica de procesado.
	if (optionSelect.equals(OPTIONS_SELECT_RUIDO[0]))//Ruido gaussiano
	{
imagenModificada.ruidoGaussiano(panelParametros.getMedia(),panelParametros.getVarianza());
		cimModificada.actualiza(imgModificada);
		showStatus("Mostrando imagen modificada (Añadido Ruido Gaussiano)");
	}
```

```
                else if(optionSelect.equals(OPTIONS_SELECT_RUIDO[1]))//Ruido impulsivo
                {
                        imagenModificada.ruidoImpulsivo(panelParametros.getPorcentaje());
                        cimModificada.actualiza(imagenModificada);
                        showStatus("Mostrando imagen modificada (Añadido Ruido Impulsivo)");
                }
                showStatus("Terminado");
        }

        /**
         * Método que crea la apariencia gráfica del applet.
         * En este caso será un panel en la parte superior para que el usuario elija el tipo de ruido y
         * el tipo de filtro a aplicar junto con sus parámetros.
         * Inmediatamente debajo hay un botón 'Aplicar' para aplicar los parámetros seleccionados
         * a la imagen original.
         * En la parte inferior aparecerán 3 contenedores que mostrarán
         * la imagen original, la imagen con ruido y la imagen ruidosa  filtrada.
         */
        private void crearElementosGraficos()
        {

                showStatus("Creando la apariencia gráfica del applet");
                setLayout(new BorderLayout());

                //Creamos el panel que nos permite seleccionar tipo de técnica de procesado
                //(junto con sus parámetros) a emplear.
                panelParametros = new EjemploPanel(this);
                add(panelParametros,"North");

                //Panel con los contenedores que mostrarán las imagenes(original, modificada).
                Panel panelImg=new Panel();

                //Se crea el primer contenedor y se carga la imagen original en él.
                //Con getImagen obtenemos la imagen de la BBII
                cimOriginal=new ImagenCanvas(getImagen(1),this);
                cimOriginal.modificable=true; //Se muestran todas las opciones del menú desplegable
                cimOriginal.addObserver(this); //Para que se muestre el menú desplegable

                panelImg.add(cimOriginal);

                //Se crea el segundo contenedor sin ninguna imagen (de momento)
                cimModificada=new ImagenCanvas(null,this);
                cimModificada.modificable=false; //Sobre la img modificada no hay menú desplegable
                panelImg.add(cimModificada);

                add(panelImg, "South");
        }

        /**
         * Método que escucha los eventos del applet-->
         * Se llamará cuando se pulse el botón 'Aplicar' del applet.
         * Aplica la técnica de procesado seleccionada
         */
        public void actionPerformed(ActionEvent e)
        {
                //Lanzamos el hilo de ejecución del applet para que se llame al método run()
                filtra();
        }

}
```

Figura 4.51: Estructura interna de clases que heredan de IMAGineApplet (esas clases tiene sufijo 'Applet')

Carece de sentido explicar detalladamente cada una de las clases de los cursos, puesto que todas siguen el esquema descrito y las únicas diferencias entre ellas serán las particularidades de cada uno de los applet, según para que hayan sido diseñados (obviamente, el Panel de un applet de procesado Morfológico que sirve para aplicar una erosión a la imagen va a ser distinto al panel utilizado en el applet de adición de ruido gaussiano, ya que los parámetros a seleccionar son diferentes, pero en ambos casos va a ser un panel de selección de parámetros para la técnica de procesado a aplicar).

Tan solo explicaremos más en detalle las clases que contienen los algoritmos con la funcionalidad de algunas de las técnicas de procesado de los cursos:

Curso Procesado Básico de Imágenes:

- *FiltradoMedianaAlgoritmo.java:* implementa el algoritmo para obtener la mediana de una serie de valores al aplicar un filtro de este tipo.

Curso Procesado Morfológico:

- *EE.java:* representa un Elemento Estructurante, necesario para las técnicas de procesado morfológico. También contiene EE predefinidos.

- *ProcMorf.java:* contiene todos los algoritmos del procesado morfológico de imágenes (erosión, dilatación, apertura, cierre, hit miss, adelgazar, engordar, esqueletizar, cercar, podar, etc.).

Curso Restauración de Imágenes:

- *RestauracionImg.java:* contiene los algoritmos necesarios para los applets del curso de restauración de imágenes y que no están implementados como técnicas de procesado de los submenús de imágenes en grises y a color.

Se han implementado los algoritmos para multiplicar píxel a píxel dos imágenes en grises y dos imágenes a color:

```
public int[][] multiplicaDosImagenes(Imagen img1, Imagen img2)
public ImagenColor multiplicaDosImagenes (ImagenColor img1, ImagenColor img2)
```

Algoritmo para multiplicar un array de números complejos que representa el módulo de la TF de una imagen con una imagen que representa una máscara (para el applet de Cancelación de Interferencias Sinusoidales):

```
public Array2D multiplicaDosModulos(Array2D moduloTF, ImagenColor mascara)
```

Algoritmo para distorsionar imágenes geométricamente en base a distorsiones prefijadas (pin-cushion, barrel, distorsiones verticales y horizontales de varios tipos):

```
public int[][] distorsionGeometrica(String tipoDistorsion)
```

4.2.4. APPLETS: MANUAL RÁPIDO PARA FUTURAS AMPLIACIONES DE LA HERRAMIENTA

En este apartado daremos las pautas básicas para facilitar futuras ampliaciones de la Herramienta de acuerdo con la estructura para la misma establecida en este proyecto. En muchos casos son tan solo las normas de estilo y diseño que hemos seguido, pero que la persona que esté desarrollando el código tendrá que sopesar si se adaptan a sus necesidades, aplicando sobre todo la lógica y el sentido común.

NOTA: Recomendamos montar el proyecto en Eclipse [Descarga gratuita en http://www.eclipse.org/downloads/], ya que en el directorio Web IMAGine\CodigoJava\ se ha copiado el .project y el .classpath con toda la configuración, y lo único que habría que hacer es copiar el directorio CodigoJava en alguna ruta en el ordenador y abrirlo en el Eclipse seleccionando esa carpeta desde la opcion File>Import>Existing project into Workspace>Browse (aquí seleccionar la carpeta CodigoJava copiada previamente). Así el nombre de la carpeta con los binarios no será necesario modificarlo ni configurar nada más (si se cambia la carpeta que contiene los .class por otro nombre que no sea bin, deberá cambiarse también el la clase CargaImagenesCtes para que funcione el Editor de imágenes). Este programa es uno de los más utilizados en el desarrollo de aplicaciones en Java, y contiene toda la funcionalidad que se pueda necesitar en IMAGine, por lo que no creo que plante ningún problema y el usarlo facilitará la configuración inicial del proyecto.

4.2.4.1. Añadir un nuevo curso a IMAGine

Se deberá crear un nuevo directorio que cuelgue al mismo nivel que los del resto de los cursos. Luego, en su interior habrá tantos subdirectorios como applets se añadan al curso.

Lo ideal es que se cree una clase java que cuelgue directamente del directorio principal del curso, en la cual se irán implementando en distintos métodos los algoritmos de las técnicas de procesado que se añadan al nuevo curso [a esos métodos se harán referencia desde la clase *Imagen.java* (si la técnica se aplica sobre imágenes a escala de grises) o *ImagenColor.java* (si la técnica se aplica sobre imágenes a color)] para aplicar las técnicas implementadas.

4.2.4.2. Añadir un nuevo applet con imágenes en grises a un curso

Se creará un subdirectorio (dentro del directorio del curso al que pertenezca el applet) dándole un nombre identificativo de la funcionalidad del applet. En él se crearán 4 clases que comiencen por el mismo nombre y tengan los siguientes sufijos:

- Sufijo *Ctes*: Interfaz java con las constantes que se utilizarán en el applet: normalmente, estas serán los nombres de los botones/opciones que al ser pulsados/seleccionados desencadenarán eventos (por ejemplo, que se aplique la técnica de procesado seleccionada sobre la imagen original). [Si el applet es muy sencillo o solo se utilizan técnicas del menú emergente, puede que esta clase no sea necesaria].

- Sufijo *Applet*: Clase java que hereda de *IMAGineApplet* y contiene el hilo de ejecución del applet. Debe tener la estructura interna descrita en la [Figura 4.50] con las modificaciones necesarias sobre el metodo *run()* y *creaElementosGraficos()* necesarias para que el applet contenga los elementos y la funcionalidad que queramos.

Aquí se crearán los contenedores de imágenes necesarios (*ImagenCanvas.java*) y en ellos pondremos las imágenes en escala de grises (*Imagen.java*).

- Sufijo *Panel*: Clase java que hereda de *Panel* y contiene los elementos gráficos para configurar los parámetros de la técnica de procesado a aplicar sobre la imagen (combos donde se selecciona uno u otro filtro, campos de texto para introducir valores, imagen para configurar el elemento estructurante que vamos a utilizar, etc.). Debe contener los métodos para obtener los valores seleccionados/introducidos o dibujados en dichos elementos gráficos en el momento en el que se pulse el botón Aplicar del Applet. [Si en el applet solo se utiliza el menú emergente que proporciona ImagenCanvas para las imágenes a escala de grises, esta clase no sea necesaria].

- Sufijo *Canvas*: Clase java que hereda de *Canvas* y sirve para añadir al applet algún tipo de dibujo (malla donde seleccionar puntos, forma de un filtro, módulo de la TF, etc.) [Esta clase solo será necesaria en aquellos applets que requieran de algún tipo de dibujo].

Lo más recomendable será copiar el subdirectorio de alguno de los applets con imágenes a escala de grises existentes que tenga una estructura parecida a lo que queramos hacer e ir modificando las cosas que deseamos cambiar, o partir de un applet muy sencillo al que vayamos añadiendo elementos poco a poco.

4.2.4.3. Añadir un nuevo applet con imágenes a color a un curso

Los pasos a seguir son los mismos que hemos descrito para la adición de un nuevo applet con imágenes a escala de grises a un curso.

La única diferencia radica en la clase con sufijo Applet, ya que los contenedores de imágenes serán del tipo *ImagenColorCanvas.java* y en ellos se añadirán las imágenes a color (de la clase *ImagenColor.java*).

No obstante, lo más recomendable seguirá siendo copiar el subdirectorio de alguno de los applets a color existentes que tenga una estructura parecida a lo que queramos hacer e ir modificando las cosas que deseamos cambiar, o partir de un applet muy sencillo al que vayamos añadiendo elementos poco a poco.

4.2.4.4. Añadir una técnica de procesado nueva al menú emergente de imágenes a escala de grises

Para que aparezca la técnica de procesado en el menú emergente, se deberá añadir en último lugar su nombre al array OPCIONES_MENU_EMERGENTE de la clase *ImágenesCanvasCtes.java*.

Para que al ser seleccionada se desencadene alguna acción, habrá que contemplar la nueva técnica en el escuchador de eventos de la clase *ImagenCanvas.java* (dentro del método *actionPerformed* de la subclase *EventosMenuEmergente*).

Si se necesitase de algún panel auxiliar emergente para definir algún parámetro configurable de la técnica, este se deberá crear dentro de la clase *ImagenCanvas_PanelesSubmenu.java*, de manera similar a como están creado el resto de paneles auxiliares de otras técnicas: se crea dentro una subclase con el nombre del panel con los métodos para obtener los valores seleccionados o introducidos, a los que se hace referencia desde fuera de la subclase; además, hay que contemplar este nuevo panel en el constructor de la clase *ImagenCanvas_PanelesSubmenú.java*. El nombre del botón 'Aplicar Técnica de Procesado' del panel emergente se añade a la clase *ImagenCanvasCtes.java* para contemplarlo en el escuchador de eventos de la clase *ImagenCanvas.java* para aplicar la técnica de procesado elegida con los parámetros seleccionados cuando se pulse ese botón.

La nueva técnica de procesado en sí deberá implementarse en un método en la clase *Imagen.java* (se implementará el algoritmo que modifica la imagen original). A ese método se llamará cuando se produzca el evento de selección de la técnica en el menú emergente de imágenes en escala de grises.

4.2.4.5. Añadir una técnica de procesado nueva al menú emergente de imágenes a color.

El menú emergente de imágenes a color está implementado de forma similar al de imágenes a escala de grises, por lo que las acciones a realizar serán las mismas pero sobre las clases homónimas que contienen la palabra Color. Por tanto, para que aparezca la técnica de procesado en el menú emergente de imágenes a color, se deberá añadir en último lugar su nombre al array OPCIONES_MENU_EMERGENTE de la clase *ImágenesColorCanvasCtes.java*.

Para que al ser seleccionada se desencadene alguna acción, habrá que contemplar la nueva técnica en el escuchador de eventos de la clase *ImagenColorCanvas.java* (dentro del método *actionPerformed* de la subclase *EventosMenuEmergente*).

Si se necesitase de algún panel auxiliar emergente para definir algún parámetro configurable de la técnica, este se deberá crear dentro de la clase *ImagenColorCanvas_PanelesSubmenu.java*, de manera similar a como están creado el resto de paneles auxiliares de otras técnicas: se crea dentro una subclase con el nombre del panel con los métodos para obtener los valores seleccionados o introducidos, a los que se hace referencia desde fuera de la subclase; además, hay que contemplar este nuevo panel en el constructor de la clase *ImagenCanvas_PanelesSubmenú.java*. El nombre del botón 'Aplicar Técnica de Procesado' del panel emergente se añade a la clase *ImagenColorCanvasCtes.java* para contemplarlo en el escuchador de eventos de la clase *ImagenColorCanvas.java*, para aplicar la técnica de procesado elegida con los parámetros seleccionados cuando se pulse ese botón.

La nueva técnica de procesado en sí deberá implementarse en un método en la clase *ImagenColor.java* (se implementará el algoritmo que modifica la imagen original). A ese método se llamará cuando se produzca el evento de selección de la técnica en el menú emergente de imágenes a color.

5. ESTADÍSTICAS DE TRÁFICO RECIBIDO EN IMAGine

5. ESTADÍSTICAS DE TRÁFICO RECIBIDO

Se ha considerado interesante incluir un medidor de estadísticas para controlar el uso de la aplicación y extraer información de interés sobre el uso de la aplicación (contador de visitas e histórico del número de visitas, navegador utilizado, resolución de pantalla utilizada, palabras de búsqueda y buscadores que han utilizado los usuarios para llegar a IMAGine, país de procedencia del usuario que visita la página, etc.).

Nos hemos decantado por incluir el Google Analitycs [http://www.google.com/analytics], debido a la gran cantidad de datos estadísticos que nos ofrecía. Para ello, hemos insertado en la página un pequeño script que permite a esta herramienta hacer las mediciones:

```
<script src="http://www.google-analytics.com/urchin.js" type="text/javascript"></script>
<script type="text/javascript">
        _uacct = "UA-2823541-1";
        urchinTracker();
</script>
```

Los datos obtenidos son muy relevantes:

- **Histórico de visitas:**

La web ha recibido 2751 visitas desde Enero de 2007 hasta Febrero de 2008.

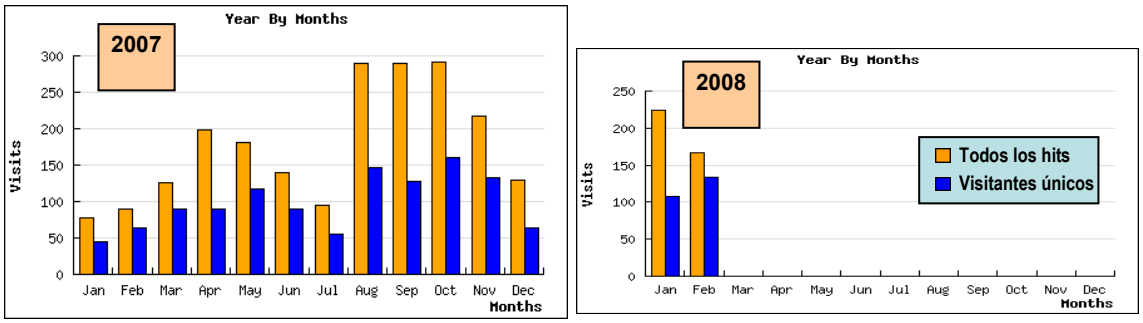


Figura 5.1: Histórico de visitas a IMAGine desde Enero 2007 hasta Febrero 2008.

Se observa que el número de visitantes únicos en Enero y Febrero de 2008 es aproximadamente el doble que en los mismos meses del año anterior. Lo que revela el aumento de la popularidad de la página desde que se subió la nueva versión de IMAGine, mucho más llamativa y completa, en Junio de 2007.

Durante los meses de septiembre a febrero la web cobra especial importancia dado que es el periodo en el que se imparte la asignatura Tratamiento Digital de Imagen en la UC3M, y esta web está pensada también como una herramienta de apoyo a la docencia, y dadas las estadísticas, parece que se estan alcanzando las expectativas.

El examen de la asignatura Tratamiento Digital de Imagen en la UC3M se realizó el 11/02/08 a las 10:00, y los días previos se registró un aumento significativo del número de visitas. Este dato revela que los estudiantes accedieron a la web para completar sus conocimientos sobre la asignatura en vísperas del examen.

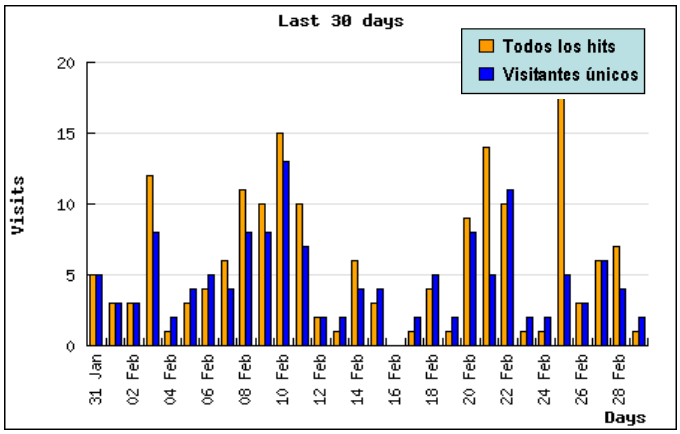


Figura 5.2: Visitas a IMAGine en Febrero 2008 (fecha examen de la asignatura).

• Países de procedencia de las visitas:

Actualmente, la mayoría de las visitas son procedentes de España y países de habla hispana (Sudamérica). La web está creada solamente en castellano, y eso sin duda limita la expansión de la popularidad de la web a países con otros idiomas (este es uno de los puntos que hemos planteado como hito para futuras expansiones de la web).

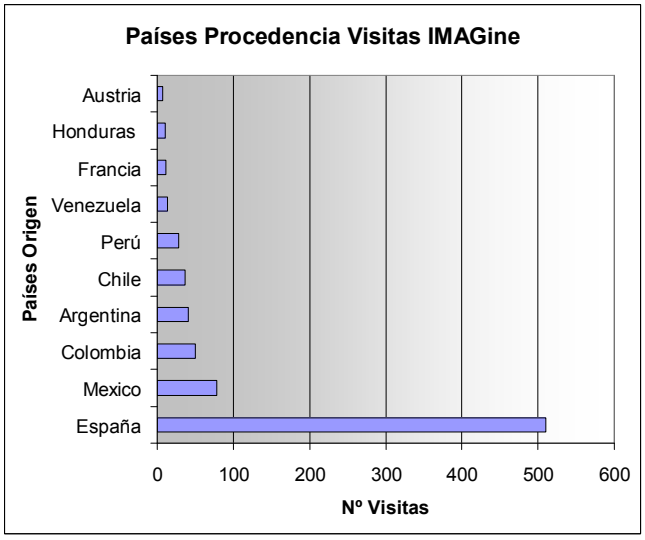


Figura 5.3: Países de procedencia de las visitas a IMAGine.

• **Navegadores utilizados:**

La web está diseñada para que se visualice correctamente al menos con Internet Explorer y Mozilla Firefox, que se consideran los navegadores más utilizados en la actualidad. Hemos tenido en cuenta las últimas versiones de estos navegadores en el diseño, pero también contemplando peculiaridades de versiones anteriores (IE 7 respecto a peculiaridades en ciertas propiedades en IE6 e inferiores, por ejemplo).

Las estadísticas de esta sección, por tanto, son muy interesantes para saber sobre que navegadores optimizar el diseño de la web, ya que cada navegador hace distinta interpretación del código html/css y esto provoca que la web se pueda ver distinta en cada uno.

Internet Explorer y Mozilla Firefox son los navegadores más utilizados para navegar por IMAGine, con bastante supremacía sobre el resto, por lo que no habrá ningun tipo de problema con la web dado que está optimizada para que se visualice correctamente en ambos.

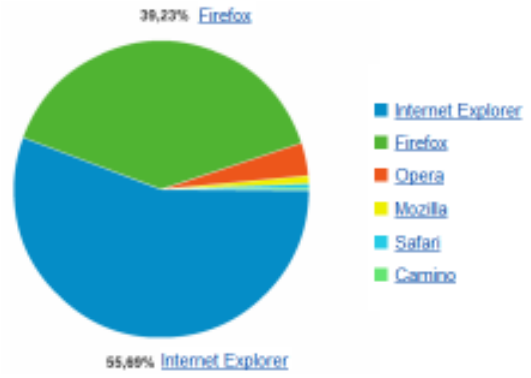


Figura 5.4: Estadísticas de navegadores utilizados en la navegación por IMAGine.

• **Resolución de pantalla utilizada:**

Es interesante conocer las resoluciones de pantalla más utilizadas para la visualización de la web. Es interesante optimizar la web para dichas resoluciones de pantalla (el tamaño de los applets cambia según la resolución, al igual que el tamaño de las fuentes y las proporciones de los menús).

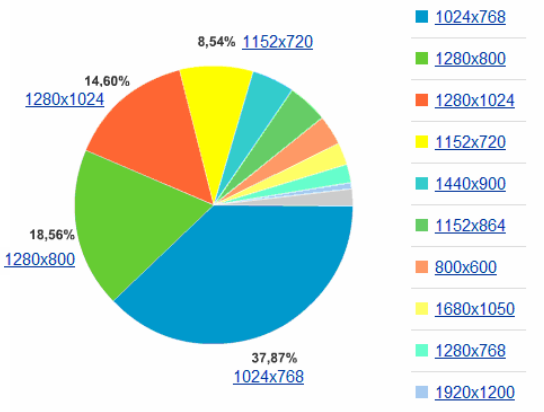


Figura 5.5: Estadísticas de resoluciones de pantalla utilizadas en la navegación por IMAGine.

• **Otras estadísticas:**

Google Analitycs ofrece multitud de estadísticas más, como la procedencia de las visitas (si han llegado a la web directamente, a traves de buscadores, por links de referencia desde otras páginas indicando cuales son esas páginas), las palabras utilizadas en los buscadores que han dado como resultado una visita a IMAGine,etc.

La velocidad de conexión es otra de las estadísticas que podemos ver, para saber cuantos usuarios podrían soportar una web más o menos pesada (tiempo que tarda en cargarse la página, visualización de videos y otros contenidos multimedia, etc.).

Velocidad de conexión		Visitas ↓
1.	DSL	287
2.	Unknown	235
3.	T1	183
4.	Cable	89
5.	Dialup	14

Figura 5.6: Estadísticas de velocidad de conexión al conectarse a IMAGine.

La estadística del número de visitantes a IMAGine que soportan aplicaciones en Java en su navegador nos revela que está en torno al 99%. Este dato es muy importante para la visualización completa de la web, ya que sino no podrían ver los applets, perdiendo una gran parte de la interactividad de la web.

Otro dato significativo es el de las palabras claves más utilizadas en búsquedas que han dado como resultado una visita a la web.

Dimensión: Palabra clave		Visitas ↓
1.	tratamiento digital de imagenes	35
2.	ejemplos de morfologia	23
3.	tratamiento digital de imagen	12
4.	ejemplo de morfologia	10
5.	imagine sueiro	10
6.	java tratamiento imagen	9
7.	morfologia binaria	8
8.	imagenes de morfologia	7
9.	tratamiento digital de la imagen	7
10.	tratamiento digital imagen	7

Figura 5.7: Palabras claves más utilizadas en búsquedas que han dirigido a IMAGine.

6. CONCLUSIONES

6. CONCLUSIONES

Este proyecto a marcado una diferencia sustancial en la línea seguida en proyectos fin de carrera anteriores relacionados con IMAGine, suponiendo un cambio total en la parte web de la aplicación (carencia principal de versiones previas de IMAGine) y una reestructuración completa en el código Java de los Applets.

Todo el diseño se ha hecho pensando en futuras ampliaciones de la herramienta, que no se tuvieron en cuenta en las primeras versiones. En ellas, no se conocía el alcance, utilidad y posibilidades que ofrecía el proyecto, y por tanto solo se pensó en hacer una pequeña aplicación que no abarcase nada más que lo implementado en la misma. Hacía falta una estructuración de la web y del proyecto en Java que implementa los applets.

Se ha creado una web completamente nueva con una estructura bien definida y una navegación fácil e intuitiva. Además, se ha cuidado la parte estética intentando hacer un diseño sencillo a la par que llamativo. Se ha tenido en cuenta los fines didácticos del proyecto, junto con la posibilidad de que el profesor de la asignatura Tratamiento Digital de Imagen pueda impartir la clase apoyándose en la propia web, por lo que se han incluido tres versiones sobre el mismo contenido (la versión web con explicaciones, applets y tamaño de letra normal para una navegación por la web desde un ordenador; la versión docente con tamaño de letra grande, applets y sin explicaciones, para que el profesor pueda impartir una clase desde ella proyectándola en una pantalla; y por último la vista de impresión, para que un usuario pueda imprimirse los contenidos teóricos sin applets ni fondos y pueda estudiarlos más comodamente sobre papel). Además, la web se ha complementado con autoevaluaciones de todos los cursos que permitan al usuario medir los conocimientos adquiridos.

Pero este proyecto no solo se ha quedado en un rediseño de todo IMAGine pensando en futuras ampliaciones y mejorando el diseño general de toda la aplicación, sino que también se ha añadido funcionalidad. Se ha creado el curso de Restauración de Imágenes y la funcionalidad de incluir imágenes a color en los applets. Además, se ha integrado la parte teórica de la asignatura (en lo concerniente a los cursos creados) en formato web, para conseguir una web más completa y mucho más navegable. Aparte, se han incluido las ya mencionadas autoevaluaciones para todos los cursos.

Con todo lo expuesto, consideramos que se han cumplido ampliamente los objetivos iniciales planteados al comienzo de este proyecto, que solo pretendía añadir el curso de Restauración de Imágenes a la web y algunas otras pequeñas mejoras, alcanzando finalmente objetivos mucho más anviciosos.

Prueba de todo ello es el elevado número de visitas que está recibiendo la web, su uso por parte de los alumnos de la asginatura Tratamiento Digital de Imágenes de la UC3M en vísperas del examen para completar sus conocimientos y la integración de la aplicación en una plataforma de e-learning desarrollada por la Universidad de Vigo, donde esperamos que sea de gran utilidad y contribuya al enriquecimiento de IMAGine.

7. LÍNEAS FUTURAS

7. LINEAS FUTURAS

Este proyecto es una continuación de proyectos fin de carrera anteriores. Esto nos hace pensar que en el futuro IMAGine se continuará ampliando con nuevos proyectos que mejoren su funcionalidad y amplíen sus contenidos.

A continuación, apuntamos algunas sugerencias sobre las líneas futuras que podrían seguir estas ampliaciones.

- **Adición de nuevos cursos a IMAGine:**

Cuando se realizó la reestructuración de IMAGine construyendo la web actual, se tuvieron en cuenta las futuras ampliaciones que probablemente sufrirá, con la adicción de nuevos curso que completen la materia impartida en la asignatura de Tratamiento Digital de Imagen en la Universidad Carlos III.

El temario de esta asignatura abarca prácticamente la totalidad de temas relacionados con el procesado de imágenes, siendo objetivo futuro de IMAGine cubrirlos todos.

1.- La imagen digital
- Percepción visual. Digitalización de imágenes. Muestreo espacial, niveles de gris. Relaciones entre píxeles: vecindad, conectividad, distancia.

2.- Procesamiento básico de imágenes. *[Ya implementado en IMAGine]*
- Operaciones aritméticas y lógicas. Transformaciones geométricas. Transformaciones de la imagen. Procesamiento espacial. Dominio de la frecuencia. Modificación del histograma. Eliminación de ruido. Realce de bordes. Falso color.

3.- Fundamentos de restauración de imágenes. *[Ya implementado en IMAGine]*
- Modelos de degradación de imágenes: imágenes antiguas, imágenes desenfocadas, imágenes con ruido. Técnicas de restauración.

4.- Codificación de imágenes.
- Cuantificadores escalares y vectoriales. Codificación por transformadas. Estándares de codificación: JPEG.

5.- Extracción de características y segmentación de imágenes.
- Detección de bordes. Texturas. Detección de movimiento. Segmentación por fronteras y por regiones. Agrupamiento (clustering)

6.- Transformaciones morfológicas *[Ya implementado en IMAGine]*
- Elementos del procesado morfológico. Dilatación. Erosión. Filtros morfológicos. Algunos algoritmos morfológicos. Extensión a imágenes de grises

7.- Descripción de objetos.
- Representación. Descriptores de frontera y de región. Descripción mediante componentes principales. Descriptores relacionales

8.- Reconocimiento de objetos.
- El clasificador bayesiano. Métodos de máxima verosimilitud. Métodos máquina.

9.- Indexado, búsqueda y recuperación de imágenes
- Indexado. Búsqueda y recuperación de imágenes. Estándares para indexación: MPEG-7.

10.- Visión estereoscópica.
    • Modelado 3D. Profundidad. El problema de la correspondencia.

11.- Biometría
    • Reconocimiento de caras. Reconocimiento del iris. Reconocimiento de huellas dactilares.

Figura 7.1: Temario asignatura Tratamiento Digital de Imagen en la Universidad Carlos III.

Los cursos que actualmente hay implementados abarcan algunos de estos temas, incluyendo en cada uno la teoría correspondiente al tema, applets demostrativos y autoevaluaciones.

- Curso Procesado Básico de Imágenes: corresponde al Tema 2.

- Curso Procesado Morfológico: corresponde al Tema 6.

- Curso Restauración de Imágenes: corresponde al Tema 3.

El resto de los cursos que se añadan deberán seguir la estructura de los existentes para conservar la homogeneidad de la aplicación.

• Ampliación de la funcionalidad de imágenes a color en los applets:

En cuanto a la parte Java de la aplicación, se ha añadido la opción de insertar en los applets imágenes a color, ofreciendo un submenú propio para ellas. Este submenú ofrece tan solo unas opciones básicas en la actualidad (mostrar el histograma, filtrado, rotación, modificar intensidad de una componente de la imagen, Transformada de Fourier, etc.).

En futuras ampliaciones, se deberían añadir nuevas opciones en el submenú de imágenes a color, incluyendo algunas de las opciones que ya estaban para las imágenes en grises, como la de adicción de ruido o el filtrado en frecuencia.

Además, se puede aprovechar las posibilidades que ofrece trabajar con imágenes a color insertando opciones nuevas que permitan, por ejemplo, realizar una ecualización del histograma por componentes (bien directamente sobre las 3 componentes del modelo de color RGB o tras realizar una transformación al modelo HSV y ecualizando solo la componente de Saturación). Otra opción que se podría incluir sería la de modificar la intensidad de un componente en base a alguna tabla de parametrización (esta es la idea que se sigue para restaurar imágenes a color deterioradas por el paso del tiempo).

- **Mostrar estadísticas en la propia web:**

Actualmente, ya se está realizando un seguimiento de las estadísticas de la web (número de visitas, navegador utilizado, país desde el que se realiza la conexión a la web) mediante la herramienta Google Analytics [http://www.google.com/analytics/es-ES/]. En el futuro se podría estudiar alguna forma de insertar estas estadísticas en la propia web de IMAGine, para que cualquier usuario las puede ver, a modo meramente informativo.

- **Ofrecer IMAGine en varios idiomas:**

La obtención de estas estadísticas también nos puede ayudar para saber si se reciben un número elevado de visitas desde países de habla no hispana, y por tanto merece la pena ofrecer una versión en inglés de la web para que esta sea accesible a un mayor público. Esta es una opción muy interesante, pero también muy costosa de realizar y de mantener, puesto que habría que reescribir todos los textos en inglés y añadir una etiqueta de lenguaje a toda la web para saber que se debe visualizar cuando este un lenguaje seleccionado y cuando otro.

- **Aumentar la interactividad de IMAGine: foros, grupos de noticias, blogs …**

Actualmente, IMAGine solo ofrece una vía de contacto entre los usuarios y el administrador de la web, que es vía e-mail. Existen multitud de elementos en el mundo del web que permiten intercambio de información entre todos los entes implicados en proceso (usuarios con usuarios, usuarios con el desarrollador web, con el administrador de la herramienta de comunicación, etc.) y que sería interesante implementar en un curso como este, que pretende ser una herramienta realmente interactiva.

Por mencionar algunos de ellos, podríamos nombrar los foros (se podría crear uno de cada curso de IMAGine), grupos de noticias relacionados con el mundo de la imagen digital, blogs con opiniones y comentarios de los usuarios, un buzón de sugerencias (con un formulario tipificado para depositarlas), etc.

- **Crear una herramienta de restauración automática de imágenes:**

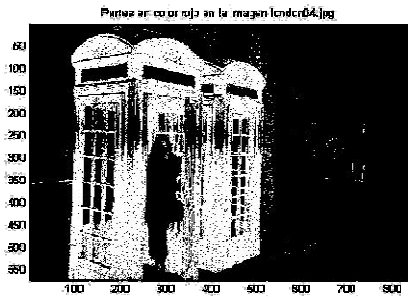
Por último, en lo referente al tema de Restauración de Imágenes, y quizá fuera de lo que englobaría IMAGine (que es un curso de orientación más didáctica que de aplicación a casos reales), se podría diseñar una herramienta que permitiese restaurar imágenes de cualquier tipología (que no obedezcan a un patrón concreto) de manera genérica. Es decir, dada una imagen real, la herramienta tendría que analizar si sufre cualquier tipo de degradación, y de ser así, estimar cual es. Tras esa estimación, se deben emplear las técnicas de restauración apropiadas para corregir los defectos de esa imagen. Por ejemplo, si se analiza y se estima que sufre una degradación de tipo desenfoque y que la imagen además presenta ruido, se deberá aplicar un filtro de Wiener. Si la degradación que sufre la imagen solo afecta al contraste y es una imagen en blanco y negro, se hará una ecualización del histograma. Así se procederá sucesivamente para el resto de casos en los que se estime que la imagen es susceptible de ser restaurada.

- **Mejorar y ampliar las autoevaluaciones:**

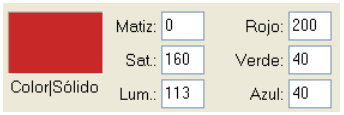
Actualmente las autoevaluaciones son únicamente formularios con preguntas y un grupo de posibles respuestas. Este formato es práctico y sencillo, pero se podría mejorar considerablemente con herramientas más sofisticadas que ampliasen el horizonte de posibilidades para el tema de las autoevaluaciones.

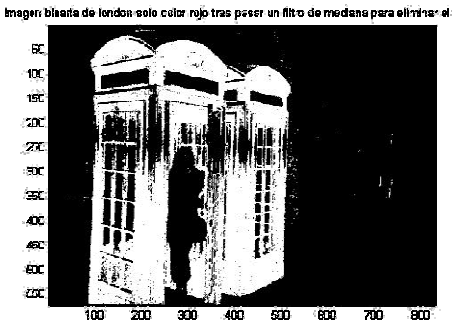
Una sugerencia podría ser un applet de autoevaluación, que permitiese plantear de manera aleatoria un problema al usuario y él dinámicamente tuviese que ir solucionándolo empleando las distintas herramientas que le proporcionase el applet hasta obtener el resultado deseado (una calidad de la imagen suficiente, el número de células no conectadas de una imagen, las partes de una imagen que tienen tonalidad roja obtenidas como una imagen binaria donde la tonalidad roja sea blanco y el resto de tonalidades negro…).

Imagen original: *Nuestro objetivo es obtener una imagen binaria donde en blanco sean las tonalidades rojas de esta imagen, y el negro el resto de tonalidades. La tonalidad roja está caracterizado por mucha intensidad en la componente R y muy poca o nada en la G y B.*

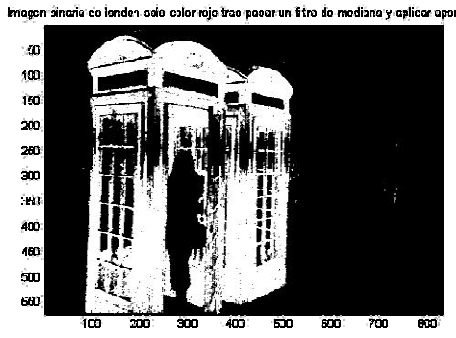


1) *Obtenemos las regiones en color rojo de la imagen aplicando el operador umbral para convertirla en una imagen binaria a partir del espacio RGB (si un píxel tenía un valor muy alto en la componente R y bajo en la G y en la B eso quería decir que el píxel era de color rojo). Vemos que en efecto aparecen resaltadas las cabinas cuyo color real es el rojo.*





2) *Imagen 1 tras aplicar un filtro de mediana para eliminar ruido*



3) *Imagen tras aplicar filtro mediana y una apertura con un EE de tipo disco de anchura 1.*

Figura 7.2: Ejemplo práctico de un posible ejercicio para el applet de autoevaluación.

8. ANEXOS

8. ANEXOS 161

8.1. CLASES MÁS IMPORTANTES DE LA AMPLIACIÓN A IMÁGENES A COLOR_ 161

8.1.1. Clase ImagenColor 161
8.1.2. Clase ImagenColorCanvasCtes 170
8.1.3. Clase ImagenColorCanvas 171
8.1.4. Clase FFTImagenColor 178
8.1.5. Clase CargaImagenesCtes 183
8.1.6. Clase CargaImagenes 184
8.1.7. Clase Editor_BaseImgColor 189

8.2. CLASES MÁS IMPORTANTES DEL CURSO RESTAURACIÓN DE IMÁGENES 196

8.2.1. Clase Imagen 196
8.2.2. Clase RestauracionImg 209

8. ANEXOS

8.1. CLASES MÁS IMPORTANTES DE LA AMPLIACIÓN A IMÁGENES A COLOR

8.1.1. Clase *ImagenColor*

```
/**
 * Clase que representa una imagen (objeto Image de Java) como un objeto ImagenColor que
 * ofrece métodos para procesar dicha imagen y obtener información/propiedades/características
 * de la misma  (por ejemplo los histogramas).
 * Como procesado se incluye la opción de filtrado, reescalado, rotación, etc.
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public class ImagenColor implements Serializable, ImagenColorCanvasCtes
{
	/* ********************************************************************************
	 *                          ATRIBUTOS
	 ******************************************************************************** */
	/**
	 * El array en el que se almacenan los pixels de la imagen.
	 * Cada pixel tiene 32-bits que se dividen en grupos de cuatro
	 * octetos de bits, o bytes. Tres de estos grupos representan
	 * la cantidad de colores fundamentales:
	 * rojo, verde y azul, que forman parte del color del pixel;
	 * y el cuarto byte, normalmente conocido con byte alpha, se utiliza
	 * para representar la transparencia, en un rango de 0 a 255, siendo 0
	 * la transparencia total y 255 la opacidad completa.
	 * (En teoría, este formato permite la utilización de 16 millones
	 * de colores en cada pixel individual).
	 * En esta representación bidimensional del array la primera dimension
	 * corresponde a la posición en el eje de las x del pixel, y la segunda
	 * a su posición en el eje de la y
	 */
	private int pixelsBi[][];

	/**
	 * El array en el que se almacenan los pixels de la imagen.
	 * Cada pixel tiene 32-bits que se dividen en grupos de cuatro
	 * octetos de bits, o bytes. Tres de estos grupos representan
	 * la cantidad de colores fundamentales:
	 * rojo, verde y azul, que forman parte del color del pixel;
	 * y el cuarto byte, normalmente conocido con byte alpha, se utiliza
	 * para representar la transparencia, en un rango de 0 a 255, siendo 0
	 * la transparencia total y 255 la opacidad completa.
	 * (En teoría, este formato permite la utilización de 16 millones
	 * de colores en cada pixel individual).
	 */
	private int pixelsUni[];

	/**
	 * El ancho en pixels de la imagen.
	 */
	private int w;

	/**
	 * El alto en pixels de la imagen.
	 */
	private int h;

	/**
	 * El titulo de la imagen. Puede modificarse segun se vayan realizando
	 * operaciones.
	 */
	private String titulo ="sin titulo";

	/**
	 * Posición Rojo (R), Verde (G), Azul (B) y byte Alpha-transparencia
	 * dentro de el array que representa el valor de un pixel
	 */
	private static final int POS_R=0;
	private static final int POS_G=1;
	private static final int POS_B=2;
	private static final int POS_ALPHA=3;
	/**
	 * Objeto image que contiene la imagen cargada de la base de datos
```

```
 */
private Image image;

/**
 * Objeto que sirve para realizar operaciones con el objeto image
 */
private BufferedImage bImage;


/* ************************************************************************
 *                        CONSTRUCTORES
 ************************************************************************ */

/**
 * Constructor de la clase que carga una imagen a partir de su nombre
 * @param nombreImage que queremos cargar
 */
public ImagenColor (String nombreImage)
{
        CargaImagenes.loadImagenColor(nombreImage);
}

/**
 * Constructor de la clase que carga la imagen que está
 * en la posición indicada por parámetros en la base de imágenes
 * del servidor
 * @param numImage posición de la imagen que queremos cargar en
 * la base de imágenes del servidor
 */
public ImagenColor (int numImagen)
{
        CargaImagenes.loadImagenColor (numImagen);
}

/**
 * Carga la imagen contenida en el objeto Image que
 * pasamos por parámetros al método.
 * (Es el constructor principal. Los demas constructores al final
 * llaman a este)
 * @param image Objeto Image que representa la imagen que mostramos
 */
public ImagenColor (Image image)
{
        this.image = image;
        inicializaAtributos(image);
}

/**
 * Crea una imagenColor como copia de otra
 * @param imagenColor que queremos copiar.
 */
public ImagenColor (ImagenColor imagenColor)
{
        this.image = imagenColor.getImage();
        this.titulo = imagenColor.getTitulo();
        inicializaAtributos(image);
}

/* ************************************************************************
 * MÉTODOS AUXILIARES DE LOS CONSTRUCTORES PARA
 * INICIALIZAR ATRIBUTOS DE LA CLASE
 ************************************************************************ */
/**
 * Inicializa los atributos de la clase una vez obtenido
 * el objeto Image que contiene la imagen cargada
 * @param image objeto Image que contiene la imagen cargada
 */
private void inicializaAtributos(Image image)
{
        this.bImage= FuncionesImage.creaBufferedImage(image);
        this.h = image.getHeight(null);
        this.w = image.getWidth(null);

        // Se declara un array para guardar la representación de la imagen
        // en pixels individuales
        this.pixelsUni = FuncionesImage.getPixels(image);
        this.pixelsBi=FuncionesImage.getArrayPixelsBidimensional (w,h,pixelsUni);

}
/* ************************************************************************
 *          MÉTODOS PARA OBTENER/CAMBIAR LOS
```

```
 *                          ATRIBUTOS DE LA CLASE
 ********************************************************************************** */

/**
 * Obtiene el objeto Image con la imagen
 * @return objeto image que contiene la imagen
 */
public Image getImage()
{
        return image;
}


/**
 * Devuelve el BufferedImage que represena a la imagen
 * @return objeto BufferedImage que representa a la imagen
 */
public BufferedImage getBufferedImagen ()
{
        return this.bImage;
}

/**
 * Devuelve el array bidimensional de enteros que representa a la imagen
 * @return array bidimensional de enteros que forman la imagen
 */
public int[][] getPixelsBidimensional()
{
        return pixelsBi;
}

/**
 * Devuelve el array unidimensional de enteros que representa a la imagen
 * @return array unidimensional de enteros que forman la imagen
 */
public int[] getPixelsUnidimensional()
{
        return pixelsUni;
}

/**
 * Devuelve la anchura de la imagen
 * @return anchura de la imagen cargada en el objeto Image
 */
public int getWidth()
{
        return w;
}

/**
 * Devuelve la altura de la imagen
 * @return altura de la imagen cargada en el objeto Image
 */
public int getHeight()
{
        return h;
}

/**
 * Obtiene el título de la imagen actual
 * @return título de la imagen actual
 */
public String getTitulo()
{
        return titulo;
}


/**
 * Cambia el título de la imagen actual por el que pasámos por parámetros
 * @param titulo -> nuevo título para la imagen actual
 */
public void setTitulo(String titulo)
{
        this.titulo=titulo;
}



/* ********************************************************************************
 *                                  PIXELS
```

```
 ********************************************************************************* */

/* ***** METODOS ESTATICOS PARA TRABAJAR CON PIXELES DE IMAGENES EN COLOR  **** */

/**
 * Usamos un entero para codificar los tres colores primarios (rojo, verde, azul)
 * y el componente alpha (el nivel de transparencia).
 * Cada componente se codifica en 8 bits (azul 0-7, verde 8-15, rojo 16-23, alpha 24-31).
 * Por tanto el valor de un pixel sera un entero que llamaremos rgb
 *
 * Este método obtiene el valor del pixel en formato array a partir del
 * valor del entero rgb para tener separada en cada posición del array
 * el valor de las componentes R,G,B y Alpha
 * @param rgb valor que toma un pixel de una Image en un punto
 * @return array con el valor rgb del pixel desglosado en cada una
 * de sus componentes (R,G,B y Alpha)
 */
public static int [] getComponentesDeUnPixel (int rgb)
{
	//En este array,
	//  -la posición 0 es el rojo
	//	-la posición 1 es el verde
	//	-la posición 2 es el azul
	//	-la posicion 3 es alpha (transparencia)

	Color color = new Color(rgb);

	int [] componentesPixel = new int [4];

	componentesPixel [POS_R] = color.getRed();	//(rgb >>16 ) & 0xFF;
	componentesPixel [POS_G] = color.getGreen();	//(rgb >> 8 ) & 0xFF;
	componentesPixel [POS_B] = color.getBlue();	//rgb & 0xFF;
	componentesPixel [POS_ALPHA] = color.getAlpha();  //(rgb >>24 ) & 0xFF;

	return componentesPixel;
}

/**
 * Usamos un entero para codificar los tres colores primarios (rojo, verde, azul)
 * y el componente alpha (el nivel de transparencia).
 * Cada componente se codifica en 8 bits (azul 0-7, verde 8-15, rojo 16-23, alpha 24-31).
 * Por tanto el valor de un pixel sera un entero que llamaremos rgb
 *
 * Este método obtiene el valor del pixel en formato entero rgb a partir
 * del array con es mismo valor pero desglosado en las componentes
 * R,G,B y Alpha en cada una de las posiciones del array.
 * @param array con el valor rgb del pixel desglosado en cada una
 * de sus componentes (R,G,B y Alpha)
 * @return rgb valor que toma un pixel de una Image en un punto
 */
public static int getRGBdeUnPixel (int [] componentesPixel)
{
	//	En este array,
	//  -la posición 0 es el rojo
	//	-la posición 1 es el verde
	//	-la posición 2 es el azul
	//	-la posicion 3 es alpha (transparencia)

	//Los valores de cada componente deben estar entre 0 y 255
	for (int i=0;i<componentesPixel.length;i++){
		if (componentesPixel[i]>255){
			componentesPixel[i]=255;
		}
		if (componentesPixel[i]<0){
			componentesPixel[i]=0;
		}
	}

	/** construcción de un píxel */
	int rgb = (componentesPixel[POS_ALPHA]<<24)+
				(componentesPixel[POS_R]<<16)+
				(componentesPixel[POS_G]<<8)+
				 componentesPixel[POS_B];

	return rgb;
}


/**
 * Devuelve el valor del pixel en formato entero rgb a partir
```

```
 * del valor de sus componentes RGB y Alpha por separado
 * @param rojo valor de la componente roja del pixel
 * @param verde valor de la componente verde del pixel
 * @param azul valor de la componente azul del pixel
 * @param alpha valor de la componente alpha del pixel
 * @return entero rgb con el valor del píxel
 */
public static int getRGBdeUnPixel (int rojo, int verde, int azul, int alpha)
{
		int [] componentesPixel = new int [4];
		componentesPixel [POS_R]=rojo;
		componentesPixel [POS_G]=verde;
		componentesPixel [POS_B]=azul;
		componentesPixel [POS_ALPHA]=alpha;

		int rgb = getRGBdeUnPixel(componentesPixel);

		return rgb;
}

/**
 * Obtiene el valor de la componente roja de un pixel
 * a partir del valor rgb del pixel.
 * @param valor rgb del pixel
 * @param valor de la componente R del valor RGB
 * pasado por parámetros
 */
public static int compRojoDeUnPixel (int rgb)
{
		int rojo  = (rgb >>16 ) & 0xFF;
		return rojo;
}

/**
 * Obtiene el valor de la componente verde de un pixel
 * a partir del valor rgb del pixel.
 * @param valor rgb del pixel
 * @param valor de la componente G del valor RGB
 * pasado por parámetros
 */
public static int compVerdeDeUnPixel (int rgb)
{
		int verde = (rgb >>8 ) & 0xFF;
		return verde;
}

/**
 * Obtiene el valor de la componente azul de un pixel
 * a partir del valor rgb del pixel.
 * @param valor rgb del pixel
 * @param valor de la componente B del valor RGB
 * pasado por parámetros
 */
public static int compAzulDeUnPixel (int rgb)
{
		int azul  =  rgb & 0xFF;
		return azul;
}

/**
 * Obtiene el valor de la componente Alpha(transparencia)
 * de un pixela partir del valor rgb del pixel.
 * @param valor rgb del pixel
 * @param valor de la componente Alpha (Transparencia)
 * del valor RGB pasado por parámetros
 */
public static int compAlphaDeUnPixel (int rgb)
{
		int alpha  = (rgb >>24 ) & 0xFF;
		return alpha;
}




/**
 * Cambia el valor de la componente roja de un pixel.
```

```
 * @param rgb entero rgb con el valor del pixel
 * @param newRojo valor nuevo que se va a dar a la componente roja
 * del pixel
 * @return entero rgb con el valor del pixel con su componente roja
 * modificada
 */
public static int cambiaCompRojoDeUnPixel (int rgb, int newRojo)
{
	int [] componentesPixel = getComponentesDeUnPixel (rgb);
	componentesPixel[POS_R]=newRojo;
	int rgbNew= getRGBdeUnPixel(componentesPixel);

	return rgbNew;
}

/**
 * Cambia el valor de la componente verde de un pixel.
 * @param rgb entero rgb con el valor del pixel
 * @param newVerde valor nuevo que se va a dar a la componente verde
 * del pixel
 * @return entero rgb con el valor del pixel con su componente verde
 * modificada
 */
public static int cambiaCompVerdeDeUnPixel (int rgb, int newVerde)
{
	int [] componentesPixel = getComponentesDeUnPixel (rgb);
	componentesPixel[POS_G]=newVerde;
	int rgbNew= getRGBdeUnPixel(componentesPixel);

	return rgbNew;
}

/**
 * Cambia el valor de la componente azul de un pixel.
 * @param rgb entero rgb con el valor del pixel
 * @param newAzul valor nuevo que se va a dar a la componente azul
 * del pixel
 * @return entero rgb con el valor del pixel con su componente azul
 * modificada
 */
public static int cambiaCompAzulDeUnPixel (int rgb, int newAzul)
{
	int [] componentesPixel = getComponentesDeUnPixel (rgb);
	componentesPixel[POS_B]=newAzul;
	int rgbNew= getRGBdeUnPixel(componentesPixel);

	return rgbNew;
}

/**
 * Cambia el valor de la componente alpha de un pixel.
 * @param rgb entero rgb con el valor del pixel
 * @param newAlpha valor nuevo que se va a dar a la componente alpha
 * del pixel
 * @return entero rgb con el valor del pixel con su componente alpha
 * modificada
 */
public static int cambiaCompAlphaDeUnPixel (int rgb, int newAlpha)
{
	int [] componentesPixel = getComponentesDeUnPixel (rgb);
	componentesPixel[POS_ALPHA]=newAlpha;
	int rgbNew= getRGBdeUnPixel(componentesPixel);

	return rgbNew;
}
```

```
/* ********************************************************************************
 *      MÉTODOS PARA LAS FUNCIONALIDADES DEL SUBMENU
 *                      DE IMAGENES EN COLOR
 ******************************************************************************** */

/**
 * Cambia el objeto Image de la clase para que contenga la
 * imagen actual pero en otro espacio de color.
 *
 * @param ColorSpace nuevo.Puede valer
 * ColorSpace.CS_CIEXYZ;
 * ColorSpace.CS_GRAY; Convierte a escala de grises
 * ColorSpace.CS_LINEAR_RGB;
 * ColorSpace.CS_PYCC
 * ColorSpace.CS_sRGB
 */
public void convierteAotroEspacioDeColor (int espacioDeColor)
{
            ColorConvertOp op = new ColorConvertOp(ColorSpace.getInstance(espacioDeColor), null);
            BufferedImage bImgGris = op.filter(bImage,null);
            this.image = FuncionesImage.creaImage(bImgGris);
            inicializaAtributos(image);
            setTitulo (getTitulo() + " en grises");
}

/**
 * Filtra la imagen actual con la matriz que pasemos por parámetros.
 *
 * @param anchoMatriz que hará de máscara
 * @param altoMatriz que hará de máscara
 * @param matriz que hará de filtro de la imagen (serán los coeficientes
 * de ponderácion del valor final de un pixel respecto a sus vecinos).
 */
public void filtra (int anchoMatriz, int altoMatriz, float [] matriz)
{
            BufferedImageOp op = new ConvolveOp(new Kernel(anchoMatriz,altoMatriz,matriz));
            BufferedImage bImgFiltrada = op.filter(bImage, null);
            this.image = FuncionesImage.creaImage (bImgFiltrada);
            inicializaAtributos(image);
            setTitulo ("Filtrada de " + getTitulo());
}

/**
 * Reescala la imagen actual segun el coef que pasemos por parámetros
 * Un coeficiente superior a 1 corresponde a una ampliación, un coeficiente inferior a 1 corresponde a una reducción.
 *
 * @param factorEscala -> un valor mayor que 1 amplia, y menor que 1
 * reduce el tamaño de la imagen
 */
public void reescala (double factorEscala)
{
      º     AffineTransform tx = new AffineTransform();
            tx.scale(factorEscala, factorEscala);
            AffineTransformOp op = new AffineTransformOp(tx, AffineTransformOp.TYPE_BILINEAR);
            BufferedImage bImgReescalada = new BufferedImage( (int) (bImage.getWidth() * factorEscala), (int)
   (bImage.getHeight() * factorEscala), bImage.getType());
            this.image = FuncionesImage.creaImage (op.filter(bImage, bImgReescalada));
            inicializaAtributos(image);
            setTitulo ("Reescalada de " + getTitulo());
}


/**
 * Reescala la imagen actual transformandola en el tamaño que pasamos
 * por parámetros al métodos.
 *
 * @param factorEscala -> un valor mayor que 1 amplia, y menor que 1
 * reduce el tamaño de la imagen
 */
public void reescala (int newAncho, int newAlto)
{
            this.image = image.getScaledInstance(newAncho,newAlto, Image.SCALE_AREA_AVERAGING);
            //No quitar esto porque sino con imagenes muy grandes no da tiempo a cargarse la imagen
            image.getWidth(null);
            image.getHeight(null);

            inicializaAtributos(image);
             setTitulo ("Reescalada de " + getTitulo());
}
```

```
/**
 * Gira la imagen actual tanto como radianes pasemos por
 * parámetros al método
 *
 * @param radianes angulo de rotación de la imagen
 * @param tipoInterpolacion indica el algoritmo que se
 * utilizará para la interpolación de la información que
 * falte al girar la imagen
 */
public void gira (double radianes, int tipoInterpolacion)
{
	AffineTransform tx = AffineTransform.getRotateInstance(radianes,0,0);
	AffineTransformOp op = new AffineTransformOp(tx, tipoInterpolacion);
	BufferedImage bImgGirada = op.filter(bImage, null);

	this.image = FuncionesImage.creaImage (bImgGirada);
	inicializaAtributos(image);
	setTitulo ("Girada de " + getTitulo());
}


/**
 * Calcula el histograma de la componente roja
 *
 * @return array con el histograma de la componente roja
 * donde cada posición corresponde al número de pixeles que
 * tienen la intensidad que indica el índice del array.
 */
public int[] histogramaRojo()
{
	int cumR[] = new int[256];
	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{
			cumR[compRojoDeUnPixel(pixelsBi[i][j])]++; // count number of values
		}
	}
	return cumR;
}

/**
 * Calcula el histograma de la componente verde
 *
 * @return array con el histograma de la componente roja
 * donde cada posición corresponde al número de pixeles que
 * tienen la intensidad que indica el índice del array.
 */
public int[] histogramaVerde()
{
	int cumG[] = new int[256];
	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{
			cumG[compVerdeDeUnPixel(pixelsBi[i][j])]++; // count number of values
		}
	}
	return cumG;
}

/**
 * Calcula el histograma de la componente azul
 *
 * @return array con el histograma de la componente roja
 * donde cada posición corresponde al número de pixeles que
 * tienen la intensidad que indica el índice del array.
 */
public int[] histogramaAzul()
{
	int cumB[] = new int[256];
	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{
			cumB[compAzulDeUnPixel(pixelsBi[i][j])]++; // count number of values
		}
	}
	return cumB;
}
```

```
	/**
	 * Método que cambia la intensidad de un componente para todos los píxeles de
	 * imagen actual
	 * @param componente: puede ser "rojo", "verde " o "azul" (Cambiamos la intensidad
	 * del componente R,G o B para todos los píxeles de la imagene)
	 * @param intensidad Valor que tendrá la componente pasada por parámetros en
	 * todos los píxeles de la imagen.
	 */
	public void modificaIntensidadComponente (String componente, int intensidad)
	{
		int [][] newpix = new int [h][w];

		for (int i=0; i<h; i++)
		{
			for (int j=0; j<w; j++)
			{
				int rgb = pixelsBi [i][j];
				//Vemos que componentes queremos modificar y le asignamos el
				//nuevo valor de intensidad a esa componente

				//Comp rojo
				if (componente.equals (COMPONENTES_COLOR[0]))
				{
					newpix[i][j]=cambiaCompRojoDeUnPixel(rgb,intensidad);
				}
				//Comp verde
				else if (componente.equals (COMPONENTES_COLOR[1]))
				{
					newpix[i][j]=cambiaCompVerdeDeUnPixel(rgb,intensidad);
				}
				//Comp azul
				else if (componente.equals (COMPONENTES_COLOR[2]))
				{
					newpix[i][j]=cambiaCompAzulDeUnPixel(rgb,intensidad);
				}

			}
		}

		// Ahora se utiliza el método createImage() para obtener una nueva
		// imagen a partir del array de pixels que hemos alterado
		this.image=FuncionesImage.creaImage(w,h,newpix);
		inicializaAtributos(image);
		setTitulo ("Modificada componente " + componente + " de " + getTitulo());
	}

	/**
	 * Método que obtiene la TF de la presente imagen a color
	 */
	public FFTImagenColor getTF ()
	{
		FFTImagenColor fft = new FFTImagenColor (pixelsUni,w, h, false);
		return fft;
	}


	/**
	 * Método que obtiene la inversa de la TF
	 * de una imagen que representa la TF
	 */
	public FFTImagenColor getInverseTF ()
	{
		FFTImagenColor fft = new FFTImagenColor (pixelsUni,w, h, true);
		return fft;
	}
}
```

8.1.2. Clase ImagenColorCanvasCtes

```
/**
 * Interfaz que define las constantes necesarias para
 * el Menú Desplegable de los Canvas con imágnes a color
 * y de los botones de los paneles emergentes al pulsar
 * algunas de las acciones de este submenu
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public interface ImagenColorCanvasCtes
{
	/**
	 * Opciones que se mostrarán en el menú emergente de imágenes
	 * a color.
	 */
	public static final String [] OPCIONES_MENU_EMERGENTE = {
				"Muestra Histogramas Componentes RGB",		//0
				"Convierte a Escala Grises",			//1
				"Reescala la imagen",				//2
				"Filtra la imagen",				//3
				"Gira la imagen",				//4
				"Cambia intensidad de una componente (R, G o B)",	//5
				"Transformada de Fourier",			//6
					"Cargar imagen"					//7
				};
	/**
	 * Nombres de las 3 componenntes que conforman una imagen a color (R, G y B)
	 */
	public static final String [] COMPONENTES_COLOR = {"rojo", "verde ", "azul"};

	/**
	 * Nombre del botón que aparece en el panel de reescalar una imagen a color
	 */
	public static final String BOTON_REESCALA="Reescala";

	/**
	 * Nombre del botón que aparece en el panel de filtrar una imagen a color
	 */
	public static final String BOTON_FILTRA="Aplicar Filtro";

	/**
	 * Nombre del botón que aparece en el panel de girar una imagen a color
	 */
	public static final String BOTON_GIRA="Girar Imagen";

	/**
	 * Nombre del botón que aparece en el panel de cambiar intensidad de un
	 * componente de la imagen.
	 */
	public static final String BOTON_CAMBIA_COMPONENTE="Cambia intensidad componente";

	/**
	 * Nombre del botón que aparece en el panel de cargar una imagen.
	 */
	public static final String BOTON_CARGA_IMAGEN="Cargar";

	/**
	 * Tipos de máscaras predefinidas
	 */
	public static  final String [] TIPOS_MATRIZ={
				"Paso Bajo",		//0
				"PBajo Mejora 1",	//1
				"PBajo Mejora 2",	//2

				"Paso Alto",		//3
				"PAlto Mejora 1",	//4
				"PAlto Mejora 2",	//5
				"PAlto Mejora 3",	//6
				"Laplaciano 1",		//7
				"Laplaciano 2",		//8
				"Laplaciano 3",		//9
				"Prewitt Horizontal",//10
				"Prewitt Vertical",	//11
				"Sobel Horizontal",	//12
				"Sobel Vertical"	//13
				};
}
```

8.1.3. Clase ImagenColorCanvas

```
/**
 * Contenedor de una imagen de tipo ImagenColor. Implementa un submenú
 * que se despliega al pulsar con el botón derecho del ratón sobre
 * la imagen.
 * Gestiona también la llamada a los métodos correspondiente de la clase
 * ImagenColor cuando se pulsa una de las opciones del submenú para
 * conseguir la funcionalidad descrita en dicha opción
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public class ImagenColorCanvas  extends Canvas implements ImagenColorCanvasCtes
{
    /**
     * Contenedor de la imagen
     */
    Container pappy;

    /**
     * Imagen de tipo Image
     */
    private Image image;

    /**
     * Objeto de la clase ImagenColor que contiene el objeto image
     */
     private ImagenColor imagenColor;

    /**
     * Muestra si la imagen se puede modificar en ciertos aspectos o no
     */
    public boolean modificable = true;

    /**
     * Muestra si el menú se puede desplegar o no
     */
    public boolean popable = true;

    /**
     * Menú de elección de los diferentes tratamientos de la imagen desplegable
     * al pulsar el botón derecho del ratón sobre la imagen
     */
    public PopupMenu men;

    /**
     * Ancho y alto del contenedor de la imagen
     */
    private int w, h;

    /**
     * Observador de la clase
     */
    private Observado obs;

    /**
     * Frame que contendrá posibles ventanas emergentes
     */
    private Frame fr;

    /**
     * Clase que crea paneles para que el usuario elija parámetros
     * dependiendo de la opción seleccionada en el menú emergente
     */
    private ImagenColorCanvas_PanelesSubmenu pEmergente;

    /**
     * Clase donde se muestran las imagenes de la base de imágenes
     */
    private BaseImg_CargaPanel baseCargaPanel;
```

```
/**
 * Constructor del contenedor de la imagen de tipo ImagenColor
 *
 * @param image La imagen que se quiere meter
 * @param parent Panel donde se va a contener el ImageCanvas
 */
public ImagenColorCanvas(ImagenColor imagenColor, Container parent)
{
        inicializaAtributos (imagenColor);

        this.obs = new Observado();

        this.pappy = parent;
        if(pappy != null)
        {
                pappy.validate();
        }

                // se añade un gestor de eventos de ratón al Canvas
                addMouseListener(new MenuEmergente());

                //Pintamos la imagen en el canvas
                repaint();
}

/**
 * Inicializa las variables necesarias para poder operar con la
 * Image cargada como ImagenColor.
 * @param imagenColor imagen a color de la cual queremos incializar
 * las variables necesarias para que funcionen todas las opciones
 * de procesado
 */
private void inicializaAtributos(ImagenColor imagenColor){
        if (imagenColor!=null)
        {
                this.imagenColor=imagenColor;
                this.image = imagenColor.getImage();
                // si la imagen existe se coge el ancho y el alto de la imagen original
                w = imagenColor.getWidth()+2;
                h = imagenColor.getHeight()+26;
        }
        else
        {
                // si no hay imagen se determina un ancho y un alto concreto
                w=256+2;
                h=256+26;
        }

        // se determina la dimensión del contenedor
        setSize(w, h);
}

 /**
  * Se pinta la imagen en el contenedor, si la imagen no es cuadrada se rellena
  * con color negro las bandas que sobren hasta que la imagen sea cuadrada
  * Se escribe el título de la imagen en la parte superior del contenedor
  */
 public void paint (Graphics g)
 {
        // se pinta todo el contenedor de blanco
        g.setColor(Color.white);
        g.fillRect(0,0,w,h);
        // se pinta un recuadro negro
        g.setColor(Color.black);
        g.drawRect(0,0,w-1,h-1);

        g.setColor(Color.black);

        if (image != null)
        {
                        g.drawString(imagenColor.getTitulo(),5,15);

                // se pinta la imagen sobre ese fondo de color blanco
                g.drawImage(image, 1, 25 , this);
         }
        else
        {
                g.drawString("Sin imagen",5,15);
        }
 }
```

```
/**
 * Se actualiza el contenedor con la imagenColor
 * que se introduce como parámetro
 * @param im Imagen que se quiere cargar en el contenedor
 */
public void actualiza(ImagenColor imagenColor)
{
	//Actualizamos los atritutos de la clase
	inicializaAtributos (imagenColor);

	// si no hay imagen no se carga nada
	if (image != null)
	{
		// se cambia el observador
		obs.cambio(image);
	}
	// se repinta el contenedor de la imagen en la pantalla
	repaint();
}


/**
 * Se actualiza la imagen que se introduce como parámetro en el contenedor
 *
 * @param im La imagen que se pintará en el contenedor
 */
public void update(ImagenColor imagenColor)
{
		inicializaAtributos(imagenColor);
	repaint();
}


/**
 * Se añade un observador a la clase
 *
 * @param o El nuevo observador
 */
public void addObserver(Observer o)
{
	obs.addObserver(o);
}

/**
 * Devuelve el objeto ImagenColor que contiene
 * el contenedor Canvas en ese instante
 * @return la Imagen a color
 */
public ImagenColor getImagenColor()
{
	return imagenColor;
}

/**
 * Se actualiza el contenedor con la imagenColor
 * que se introduce como parámetro
 * @param im Imagen que se quiere cargar en el contenedor
 */
public void setImagenColor(ImagenColor imagenColor)
{
	this.imagenColor=imagenColor;
	//Actualizamos los atritutos de la clase
	inicializaAtributos (imagenColor);

	// se repinta el contenedor de la imagen en la pantalla
	repaint();
}
```

```
/**
 * Clase que gestiona los eventos de ratón. Cuando se pulsa con el
 * raton sobre el Canvas que muestra la imagen desplegará un menú
 * emergente con varias opciones.
 */
class MenuEmergente extends MouseAdapter
{
	/**
	 * Escuchador de eventos del menú emergente (para cuando se seleccione
	 * alguna de las opciones del menú emergente)
	 */
	private EventosMenuEmergente gestorEventosSubmenu;

	/**
	 * Constructor que inicializa el escuchador de eventos del submenu
	 */
	public MenuEmergente ()
	{
		gestorEventosSubmenu=new EventosMenuEmergente();
	}

	/**
	 * Método que se invoca cuando se pulsa un botón del raton sobre el
	 * canvas y muestra el menú emergente
	 */
	public void mousePressed(MouseEvent e)
	{
		// si se presiona el botón derecho se muestra el menú
		//desplegable en pantalla
		if (e.isPopupTrigger())
		{
			muestraMenu(e.getX(), e.getY());
		}
	}

	/**
	 * Método que se invoca cuando se pulsa un botón del raton sobre el
	 * canvas y muestra el menú emergente
	 */
	public void mouseReleased(MouseEvent e)
	{
		// si se presiona el botón derecho se muestra el menú
		//desplegable en pantalla
		if (e.isPopupTrigger())
		{
			muestraMenu(e.getX(), e.getY());
		}
	}

	/**
	 * Muestra un menú emergente con varias opciones
	 * @param x posición en el eje de las x donde se abrirá el menú
	 * @param y posición en el eje de las y donde se abrirá el menú
	 */
	private void muestraMenu(int x, int y)
	{
		// si el menú no se puede desplegar o no hay imagen no se muestra el
		// menú
		if ((popable == false) || (image == null))
		{
			return;
		}
		else
		{
			// se crea el menú y las diferentes opciones de tratamiento
			//digital de la imagen
			men = new PopupMenu();
			MenuItem menuItem;

			for (int i=0; i<OPCIONES_MENU_EMERGENTE.length;i++)
			{
				menuItem = new MenuItem(OPCIONES_MENU_EMERGENTE[i]);
				menuItem.addActionListener(gestorEventosSubmenu);
				// si no se puede modificar la imagen ciertas opciones no
				//se pueden elegir
				if (!modificable)
				{
					menuItem.setEnabled(false);
				}
				// se añaden las opciones creadas al menú desplegable
				men.add(menuItem);
```

```
                                }
                                add(men);

                                // muestra el menú desplegable en pantalla
                                men.show(ImagenColorCanvas.this, x, y);
                        }
                }

        }


        /**
         * Clase que gestiona todos los eventos del menu emergente (se invoca
         * cada vez que se pulsa cualquiera de las opciones incluidas en el mismo).
         */
        class EventosMenuEmergente implements ActionListener
        {
                /**
                 * Manejador de eventos de la clase. Gestiona los eventos del submenú
                 * (cuando se pincha en alguna de las opciones)
                 */
                public void actionPerformed(ActionEvent e)
                {
                        //Comando que indica que opción del menú emergente ha sido seleccionada
                        String comando = e.getActionCommand();

                        // OPCIÓN MUESTRA HISTOGRAMAS DE LAS COMPONENTES RGB DE UNA IMAGEN
                        if (comando.equals(OPCIONES_MENU_EMERGENTE[0]))
                        {

                                //Panel donde añadiremos los histogramas
                                Panel pHistogramas = new Panel (new GridLayout (1,3));

                                //Creamos los histogramas de las componentes R,G y B de la imagen
                                HistogramaCanvas hcR = new HistogramaCanvas (imagenColor,"rojo",pHistogramas);
                                HistogramaCanvas hcG = new HistogramaCanvas (imagenColor,"verde",pHistogramas);
                                HistogramaCanvas hcB = new HistogramaCanvas (imagenColor,"azul",pHistogramas);

                                //Añadimos los histogramas al panel
                                pHistogramas.add(hcR);
                                pHistogramas.add(hcG);
                                pHistogramas.add(hcB);

                                //Esto es necesario para que se muestren los histogramas ya que estamos añadiendo
                                //objetos Canvas y si no no se visualizan
                                pHistogramas.validate();
                                pHistogramas.setVisible(true);

                                //Mostramos una ventana emergente con los histogramas
                                sacaVentana("Histogramas de las componentes R, G y B", pHistogramas);
                        }

                        // OPCIÓN CONVIERTE IMAGEN A ESCALA GRISES
                        else if (comando.equals(OPCIONES_MENU_EMERGENTE[1]))
                        {
                                //Creamos la imagen en escala de grises
                                imagenColor.convierteAotroEspacioDeColor(ColorSpace.CS_GRAY);
                                //Mostramos la imagen en el canvas
                                actualiza(imagenColor);
                        }

                        // OPCIÓN REESCALA LA IMAGEN:
                        else if (comando.equals(OPCIONES_MENU_EMERGENTE[2]))
                        {
                                pEmergente = new ImagenColorCanvas_PanelesSubmenu
(OPCIONES_MENU_EMERGENTE[2],this, null);
                                sacaVentana ("Reescala una imagen", pEmergente);
                        }

                        // CONFIRMACIÓN REESCALADO DE LA IMAGEN
                        else if (comando.equals(BOTON_REESCALA))
                        {
                                //Reescalamos la imagen
                                if (pEmergente.reescala_eligeFactorEscala())
                                {
                                        //Reescalamos la imagen mediante un factor de escala
                                        imagenColor.reescala(pEmergente.reescala_getFactorEscala());
                                }
```

```
                        else
                        {
                                //Reescalamos la imagen introduciendo las dimensiones de la imagen resultante
    imagenColor.reescala(pEmergente.reescala_getAnchoImgNew(),pEmergente.reescala_getAltoImgNew());
                        }

                        //Mostramos la imagen en el canvas
                        actualiza (imagenColor);
                        quitaVentana();
                        pEmergente=null;
                }

                 // OPCIÓN FILTRA LA IMAGEN:
                else if (comando.equals(OPCIONES_MENU_EMERGENTE[3]))
                {
                        pEmergente = new ImagenColorCanvas_PanelesSubmenu
(OPCIONES_MENU_EMERGENTE[3],this, null);
                        //Se saca una ventana donde el usuario elige el tipo de filtro a aplicar
                        sacaVentana ("Selecciona el tipo de filtro a aplicar", pEmergente);
                }

                // CONFIRMACIÓN FILTRADO IMAGEN:
                else if (comando.equals(BOTON_FILTRA))
                {
                        imagenColor.filtra(3,3,pEmergente.filtro_getMatriz());//La matriz de filtrado sera de 3x3
                        actualiza (imagenColor);
                        //Se cierra la ventana donde se muestra el tipo de filtro
                        quitaVentana();
                        pEmergente=null;
                }

                // OPCIÓN GIRA LA IMAGEN:
                else if (comando.equals(OPCIONES_MENU_EMERGENTE[4]))
                {
                        pEmergente = new ImagenColorCanvas_PanelesSubmenu
(OPCIONES_MENU_EMERGENTE[4],this, null);
                        //Se saca una ventana donde el usuario elige grados del giro
                        //y tipo de interpolación
                        sacaVentana ("Elige los grados a girar y el tipo de interpolación", pEmergente);
                }

                // CONFIRMACIÓN GIRA LA IMAGEN:
                else if (comando.equals(BOTON_GIRA))
                {
                        imagenColor.gira(pEmergente.gira_getRadianes(),pEmergente.gira_getInterpolacion());
                        actualiza(imagenColor);
                        //Se cierra la ventana donde se elige el giro
                        quitaVentana();
                        pEmergente=null;
                }

                // OPCIÓN CAMBIA INTENSIDAD COMPONENTE:
                else if (comando.equals(OPCIONES_MENU_EMERGENTE[5]))
                {
                        pEmergente = new ImagenColorCanvas_PanelesSubmenu
(OPCIONES_MENU_EMERGENTE[5],this, null);
                        //Se saca una ventana donde el usuario elige a que componente
                        //quiere cambiar la intensidad y que valor quiere poner a esta
                        sacaVentana ("Cambia intensidad de una componente (R,G o B)", pEmergente);
                }

                // CONFIRMACIÓN CAMBIA INTENSIDAD COMPONENTE:
                else if (comando.equals(BOTON_CAMBIA_COMPONENTE))
                {

    imagenColor.modificaIntensidadComponente(pEmergente.cambiaComp_getComponenteSelect(),pEmergente.cambiaCo
mp_getIntensidad());
                        actualiza(imagenColor);
                        //Se cierra la ventana donde se elige el giro
                        quitaVentana();
                        pEmergente=null;
                }

                // OPCIÓN TRANSFORMADA FOURIER
                else if (comando.equals(OPCIONES_MENU_EMERGENTE[6]))
                {
                        pEmergente = new ImagenColorCanvas_PanelesSubmenu
(OPCIONES_MENU_EMERGENTE[6],null, imagenColor.getTF());
                        //Se saca una ventana con las componentes de la TF (modulo, fase...)
                        sacaVentana ("TF de " + imagenColor.getTitulo(), pEmergente, true);
```

```
                }

                // OPCIÓN CARGAR IMAGEN
                else if (comando.equals(OPCIONES_MENU_EMERGENTE[7]))
                {
                        baseCargaPanel = new BaseImg_CargaPanel(this,true);
                        // se muestra en pantalla la imagen seleccionada
                        sacaVentana("Cargar Imagen", baseCargaPanel, true);
                }

                // CONFIRMACIÓN CARGAR IMAGEN
                else if (comando.equals(BOTON_CARGA_IMAGEN))
                {
                        // se obtiene la imagen que se ha seleccionado si se ha seleccionado
                        // alguna
                        if (baseCargaPanel.getImagenColor() != null)
                        {
                                imagenColor = baseCargaPanel.getImagenColor();
                        }
                        // se quita la ventana del menú de elección de la imagen
                        quitaVentana();
                        actualiza(imagenColor);
                        baseCargaPanel = null;
                }
        }
}

//******** MÉTODOS AUXILIARES PARA MOSTRAR LAS VENTANAS EMERGENTES **********

/**
 * Abre una nueva ventana mostrando el componente p
 *
 * @param tit Título de la ventana
 * @param p Componente que se va a mostrar en la ventana
 * @param b Muestra si se puede cambiar el tamaño de la imagen
 * @param po Punto donde se va a mostrar la ventana
 */
public void sacaVentana(String tit, Component p, boolean b, Point po)
{
        // se crea la ventana
        fr = new Frame(tit);
        fr.setResizable(b);
        // se añade el gestor de eventos de ventana
        fr.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){
                Window w = e.getWindow();
                w.removeAll();
                w.dispose();
                }
});

        fr.add(p);
        fr.pack();

        // se determina la dimensión y la posición de la ventana
        Dimension e = getSize();
        Dimension d = fr.getSize();
        int a = po.x + e.width / 2 - d.width / 2;
        int c = po.y + e.height / 2 - d.height / 2;
        fr.setLocation(a > 0 ? a : 0, c > 0 ? c : 0);
        fr.show();
}

/**
 * Abre una nueva ventana mostrando el componente p
 *
 * @param tit Título de la ventana
 * @param p Componente que se va a mostrar en la ventana
 * @param po Punto donde se va a mostrar la ventana
 */
public void sacaVentana(String tit, Component p, Point po)
{
        sacaVentana(tit, p, false, po);
}
```

```
    /**
     * Abre una nueva ventana reescalable mostrando el componente p
     *
     * @param tit Título de la ventana
     * @param p Componente que se va a mostrar en la ventana
     * @param b Muestra si se puede cambiar el tamaño de la imagen
     */
    public void sacaVentana(String tit, Component p, boolean b)
    {
        sacaVentana(tit, p, b, getLocationOnScreen());
    }

    /**
     * Abre una nueva ventana mostrando el componente p
     *
     * @param tit Título de la ventana
     * @param p componente que se va a mostrar en la ventana
     */
    public void sacaVentana(String tit, Component p)
    {
        sacaVentana(tit, p, false);
    }


    /**
     * Quita la ventana emergente que pudiese haber abierta
     */
    public void quitaVentana ()
    {
        if (fr!=null)
        {
            //Se cierra la ventana donde se muestra el tipo de filtro
            fr.removeAll();
            fr.dispose();
            fr = null;
        }
    }

}
```

8.1.4. Clase FFTImagenColor

```
/**
 * La clase <code>FFTImagenColor</code> implementa la transformada de Fourier de
 * una <code>ImagenColor</code>, así como su inversa. De la FFT se obtienen objetos
 * Image con su módulo, fase, parte real y parte imaginaria. De la IFFT se obtiene
 * la Image que representa la imagen original a partir de su FFT.
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public class FFTImagenColor implements FFTImagenCtes
{

    /**
     * Dimensiones de la imagen
     */
    public int w, h;

    /**
     * Estructura de datos de entrada a el algoritmo.
     */
    public Array2D input;

    /**
     * Estructura de datos con los resultados intermedios del algritmo (después de
     * haber aplicado la 1D FFT de las columnas pero antes de aplicar también la
     * de las filas)
     */
    public Array2D intermediate;

    /**
     * Estructura de datos de salida del algoritmo (que representarán la FFT en caso
     * de haber elegido ese algoritmo o la IFFT si se ha elegido hacer la inversa de la
     * Tranformada de Fourier).
     */
    public Array2D output;
```

```
/**
 * Crea la FFT de la imagen representada por el array de pixels
 * que se pasa por parametros. O la IFFT de un array de pixels que
 * representen la FFT de una imagen
 * @param ImagenColor de la que se quiere obtener la FFT o la IFFT
 * @param inverseFFT  a true si se quiere calcular la IFFT y a false si se
 * quiere calcular la FFT
 */
public FFTImagenColor (ImagenColor imagenColor, boolean inverseFFT)
{
	this.w=imagenColor.getWidth();
	this.h=imagenColor.getHeight();
	Array2D  aux = new Array2D(imagenColor.getPixelsUnidimensional(), w, h);
	if (inverseFFT)
	{
		getInverseFFT(aux);
	}
	else
	{
		getFFT(aux);
	}
}


/**
 * Crea la FFT de la imagen representada por el array de pixels
 * que se pasa por parametros. O la IFFT de un array de pixels que
 * representen la FFT de una imagen
 * @param pixels array de enteros que representan la imagen (en caso de FFT)
 * o array que representa la FFT de una imagen (en caso de IFFT)
 * @param w anchura de la imagen en pixels
 * @param h alturo de la imagen en pixels.
 * @param inverseFFT  a true si se quiere calcular la IFFT y a false si se
 * quiere calcular la FFT
 */
public FFTImagenColor (int[] pixels, int w, int h, boolean inverseFFT)
{
	this.w=w;
	this.h=h;
	Array2D  aux = new Array2D(pixels, w, h);

	if (inverseFFT)
	{
		getInverseFFT(aux);
	}
	else
	{
		getFFT(aux);
	}
}

/**
 * Crea la FFT de la imagen representada por el array de pixels
 * que se pasa por parametros. O la IFFT de un array de pixels que
 * representen la FFT de una imagen
 * @param array que representan la imagen (en caso de FFT)
 * o array que representa la FFT de una imagen (en caso de IFFT)
 * @param inverseFFT  a true si se quiere calcular la IFFT y a false si se
 * quiere calcular la FFT
 */
public FFTImagenColor (Array2D array, boolean inverseFFT)
{
	this.w=array.width;
	this.h=array.height;

	if (inverseFFT)
	{
		getInverseFFT(array);
	}
	else
	{
		getFFT(array);
	}
}
```

```
/**
 * Crea la FFT como copia de la FFT que pasemos por parametros
 * @param fft que queremos copiar
 */
public FFTImagenColor (FFTImagenColor fft)
{
	this.w=fft.w;
	this.h=fft.h;
	this.input= fft.input;
	this.intermediate= fft.intermediate;
	this.output= fft.output;
}


/* ********************* FFT *************************** */

/**
 * Metodo que aplica recursivamente la 1D FFT a un array de NumComplejo
 *
 * @param x   Array de NumComplejo que contiene los datos de
 * las filas o columnas de una imagen
 * @return un array de NumComplejo con el resultado de la 1D FFT
 */
private NumComplejo[] recursiveFFT (NumComplejo[] x)
{
	NumComplejo z1, z2, z3, z4, tmp, cTwo;
	int n = x.length;
	int m = n / 2;
	NumComplejo[] result = new NumComplejo[n];
	NumComplejo[] even = new NumComplejo[m];
	NumComplejo[] odd = new NumComplejo[m];
	NumComplejo[] sum = new NumComplejo[m];
	NumComplejo[] diff = new NumComplejo[m];
	cTwo = new NumComplejo(2, 0);
	if (n == 1)
	{
		result[0] = x[0];
	}
	else
	{
		z1 = new NumComplejo(0.0, -2 * (Math.PI) / n);
		tmp = NumComplejo.cExp(z1);
		z1 = new NumComplejo(1.0, 0.0);
		for (int i = 0; i < m; ++i)
		{
			z3 = NumComplejo.cSum(x[i], x[i + m]);
			sum[i] = NumComplejo.cDiv(z3, cTwo);

			z3 = NumComplejo.cDif(x[i], x[i + m]);
			z4 = NumComplejo.cMult(z3, z1);
			diff[i] = NumComplejo.cDiv(z4, cTwo);

			z2 = NumComplejo.cMult(z1, tmp);
			z1 = new NumComplejo(z2);
		}
		even = recursiveFFT(sum);
		odd = recursiveFFT(diff);

		for (int i = 0; i < m; ++i)
		{
			result[i * 2] = new NumComplejo(even[i]);
			result[i * 2 + 1] = new NumComplejo(odd[i]);
		}
	}
	return result;
}

/**
 * Método que aplica la 2D FFT aplicando recursivamente la 1D FFT de las
 * filas y las columnas del array de NumComplejo que representa la imagen
 *
 * @param array que representa la imagen.
 */
private void getFFT(Array2D input)
{
	this.input=input;
	this.intermediate = input;
	this.output = input;
```

```
        for (int i = 0; i < input.size; ++i)
        {
                this.intermediate.putColumn(i, recursiveFFT(input.getColumn(i)));
        }
        for (int i = 0; i < intermediate.size; ++i)
        {
                this.output.putRow(i, recursiveFFT(intermediate.getRow(i)));
        }
        for (int j = 0; j < output.values.length; ++j)
        {
                for (int i = 0; i < output.values[0].length; ++i)
                {
                        this.intermediate.values[i][j] = output.values[i][j];
                        this.input.values[i][j] = output.values[i][j];
                }
        }
}


/* ********************* INVERSE FFT ************************** */

/**
 * Metodo que aplica recursivamente el algoritmo 1D Inverse FFT a un array
 * de NumComplejo
 *
 * @param x   Array de NumComplejo que contiene los datos de
 * las filas o columnas de la FFT de una imagen
 * @return un array de NumComplejo con el resultado de la 1D Inverse FFT
 */
private NumComplejo[] recursiveInverseFFT(NumComplejo[] x)
{
        NumComplejo z1, z2, z3, z4, tmp, cTwo;
        int n = x.length;
        int m = n / 2;
        NumComplejo[] result = new NumComplejo[n];
        NumComplejo[] even = new NumComplejo[m];
        NumComplejo[] odd = new NumComplejo[m];
        NumComplejo[] sum = new NumComplejo[m];
        NumComplejo[] diff = new NumComplejo[m];
        cTwo = new NumComplejo(2, 0);
        if (n == 1)
        {
                result[0] = x[0];
        }
        else
        {
                z1 = new NumComplejo(0.0, 2 * (Math.PI) / n);
                tmp = NumComplejo.cExp(z1);
                z1 = new NumComplejo(1.0, 0.0);
                for (int i = 0; i < m; ++i)
                {
                        z3 = NumComplejo.cSum(x[i], x[i + m]);
                        sum[i] = NumComplejo.cDiv(z3, cTwo);

                        z3 = NumComplejo.cDif(x[i], x[i + m]);
                        z4 = NumComplejo.cMult(z3, z1);
                        diff[i] = NumComplejo.cDiv(z4, cTwo);

                        z2 = NumComplejo.cMult(z1, tmp);
                        z1 = new NumComplejo(z2);
                }
                even = recursiveInverseFFT(sum);
                odd = recursiveInverseFFT(diff);

                for (int i = 0; i < m; ++i)
                {
                        result[i * 2] = new NumComplejo(even[i]);
                        result[i * 2 + 1] = new NumComplejo(odd[i]);
                }
        }
        return result;
}
```

```
/**
 * Método que aplica la 2D inverse FFT aplicando recursivamente la 1D inverse FFT
 * de las filas y las columnas del array de NumComplejo que representa la FFT de
 * una imagen
 *
 * @param array que representa la FFT de una imagen.
 */
private void getInverseFFT(Array2D input)
{
	this.input = input;
	this.intermediate = new Array2D(input.width, input.height);
	this.output = new Array2D(input.width, input.height);

	for (int i = 0; i < input.size; ++i)
	{
		intermediate.putColumn(i, recursiveInverseFFT(input.getColumn(i)));
	}

	for (int i = 0; i < intermediate.size; ++i)
	{
		output.putRow(i, recursiveInverseFFT(intermediate.getRow(i)));
	}
}


/* ************************  METODOS AUXILIARES ************************** */

/**
 * Devuelve una image con la parte real de la FFT
 * @return image que representa la parte real de la FFT
 */
public Image getReal()
{
	double[] real= Array2D.abs(output.DCToCentre(output.getReal()));
	double[] realLogs= Array2D.logs(real);
	int [] realImagen = Array2D.toPixels(realLogs);
	return FuncionesImage.creaImage(w,h,realImagen).getScaledInstance(ANCHO_FFT_IMGCOLOR,
                                                                      ALTO_FFT_IMGCOLOR,
                                                                      Image.SCALE_AREA_AVERAGING);
}

/**
 * Devuelve una image con la parte imaginaria de la FFT
 * @return image que representa la parte imaginaria de la FFT
 */
public Image getImaginary()
{
	double[] imaginary= Array2D.abs(output.DCToCentre(output.getImaginary()));
	double[] imaginaryLogs= Array2D.logs(imaginary);
	int [] imaginaryImagen = Array2D.toPixels(imaginaryLogs);
	return FuncionesImage.creaImage(w,h,imaginaryImagen).getScaledInstance(ANCHO_FFT_IMGCOLOR,
                                                                           ALTO_FFT_IMGCOLOR,
                                                                           Image.SCALE_AREA_AVERAGING);
}

/**
 * Devuelve una image con la fase de la FFT
 * @return image que representa la fase de la FFT
 */
public Image getPhase()
{
	double[] phase= Array2D.abs(output.DCToCentre(output.getPhase()));
	int [] phaseImagen = Array2D.toPixels(phase);
	return FuncionesImage.creaImage(w,h,phaseImagen).getScaledInstance(ANCHO_FFT_IMGCOLOR,
                                                                       ALTO_FFT_IMGCOLOR,
                                                                       Image.SCALE_AREA_AVERAGING);
}

/**
 * Devuelve una image con el modulo de la FFT
 * @return image que representa el modulo de la FFT
 */
public Image getMagnitude()
{
	double[] magnitude= Array2D.abs(output.DCToCentre(output.getMagnitude()));
	double[] magnitudeLogs= Array2D.logs(magnitude);
	int [] magnitudeImagen = Array2D.toPixels(magnitudeLogs);
	return FuncionesImage.creaImage(w,h,magnitudeImagen).getScaledInstance(ANCHO_FFT_IMGCOLOR,
                                                                           ALTO_FFT_IMGCOLOR,
                                                                           Image.SCALE_AREA_AVERAGING);
}
```

```
    /**
     * Devuelve una image con la imagen resultado de aplicar la IFFT de una imagen
     * en el dominio de la frecuencia
     * @return image que representa la imagen en el dominio del tiempo tras aplicar
     * la IFFT sobre una imagen en el dominio de la frecuencia
     */
    public Image getInverse()
    {
                        double [] outputArrayDoubles = output.getReal();
                        int []outputArray = Array2D.toPixels(Array2D.allPositive(outputArrayDoubles));
                        return FuncionesImage.creaImage(w,h,outputArray);
    }
}
```

8.1.5. Clase CargaImagenesCtes

```
/**
 * Interfaz que define las constantes necesarias para
 * el Applet Distorsiones Geométricas
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public interface CargaImagenesCtes
{
    /* ********************************************************************************
     *                  PARÁMETROS IMÁGENES A COLOR
     ******************************************************************************** */
    /**
     * Path de la carpeta donde se encuentran las imágenes a cargar (si es un applet)
     */
    public final static String DIR_IMAGENESCOLOR_APPLET="img/imagenes_color/";

    /**
     * Nombre de la carpeta donde se encuentran las imágenes (para cuando estas ejecutando
     * el Editor).
     */
    public final static String DIR_IMAGENESCOLOR ="bin/img/imagenes_color/";

    /**
     * Nombre de la lista donde se almacena la información de todas las imágenes
     * a color que hay disponibles
     */
    public final static String IMAGE_NAMESCOLOR = "listadoImagenesColor.txt";

    /**
     * Extensión de las imágenes
     */
    public static final String  EXT_IMG_COLOR = ".jpg";

    /**
     * Nombres de las imagenes a color que están disponibles en la Base de Imágenes del servidor
     */
    public static final String  MIN_IMG_COLOR = ".min";

    /**
     * Dimensiones que deben tener las imágenes en el panel de carga de imágenes
     * para imágenes en escala de grises
     */
    public final static int ANCHO_IMGCOLOR_CARGA=100;
    public final static int ALTO_IMGCOLOR_CARGA=100;

    /* ********************************************************************************
     *                  PARÁMETROS IMÁGENES GRISES
     ******************************************************************************** */
    /**
     * Path de la carpeta donde se encuentran las imágenes a cargar (si es un applet)
     */
    public final static String DIR_IMAGENESGRISES_APPLET = "img/imagenes_gris/";

    /**
     * Nombre de la carpeta donde se encuentran las imágenes (para cuando estas ejecutando
     * el Editor).
     */
    public final static String DIR_IMAGENESGRISES = "bin/img/imagenes_gris/";
```

```
    /**
     * Nombre de la lista donde se almacena la información de todas las imágenes
     * en escala de grises
     */
    public final static String IMAGE_NAMESGRISES = "listadoImagenesGrises.txt";

    /**
     * Extensión del archivo que contiene la imagen original (para imagenes en escala de grises)
     */
    public final static String IMAGE_EXT = ".jpg";

    /**
     * Extensión del archivo que contiene la imagen en miniatura (para imagenes en escala de grises
     */
    public final static String MIN_IMG_GRISES=".min";

    /**
     * Dimensiones que deben tener las imágenes grabadas en la base de imágenes del
     * servidor (solo para imágenes en escala de grises). No variar este parámetro
     * porque haría que algunas técnicas de procesado fallasen.
     */
    public final static int ANCHO_IMG_BASE=256;
    public final static int ALTO_IMG_BASE=256;

    /**
     * Dimensiones que deben tener las imágenes en el panel de carga de imágenes
     * para imágenes en escala de grises
     */
    public final static int ANCHO_IMGGRISES_CARGA=65;
    public final static int ALTO_IMGGRISES_CARGA=65;

}
```

8.1.6. Clase CargaImagenes

```
/**
 * La clase ImageBase carga las imágenes que están almacenadas en la base de
 * imágenes del servidor
 *
 * @author Realizado por Pablo Puente
 * @author Documentado por Fernando Hoyos
 * @author Modificado por Ana Belén Alonso Martín para incluir imágenes a color
 * @version IMAGine 2.0 - 2008
 */
public class CargaImagenes implements CargaImagenesCtes
{
    /**
     * Devuelve un array con los nombres de las imágenes que hay en la base de
     * imágenes del servidor (imágenes en escala de grises)
     *
     * @return El array con los nombres de las imágenes en escala de grises
     */
    public static String[] loadNamesImgGrises()
    {
        Vector vNames = new Vector();
        BufferedReader br=null;
        try
        {
            //Esto es para cuando es un applet
            if (IMAGineApplet.applet!=null)
            {
                //Obtenemos la ruta de las imagenes (esta en una carpeta imagenes donde se alojen los .class)
                URL rutaListado= new URL (IMAGineApplet.applet.getCodeBase()+
                                          DIR_IMAGENESGRISES_APPLET+IMAGE_NAMESGRISES);
                br =new BufferedReader(new InputStreamReader(rutaListado.openStream()));
            }

            //Esto es para cuando estamos en el Editor (no es un applet) y coge el directorio distinto
            else
            {
                br = new BufferedReader(new FileReader(DIR_IMAGENESGRISES+
                                                       IMAGE_NAMESGRISES));

            }
            String linea="";
            while ((linea=br.readLine())!=null)
            {
                vNames.add(linea);
            }
```

```
        }
        catch (Exception e)
        {
                e.printStackTrace();
        }
        finally
        {
                try
                {
                        if (br != null)
                                br.close(); // cierra la corriente de entrada de datos
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }
        }

        // Carga los nombres de las imágenes
        String namesImgColor[] = new String [vNames.size()];
        for (int i=0; i<namesImgColor.length;i++)
        {
                namesImgColor[i]=(String)vNames.get(i);
        }
        return namesImgColor;
}

/**
 * Devuelve la imagen (original) cuyo nombre se introduce como parámetro
 *
 * @param nombreImagen Nombre de la imagen que se quiere cargar
 * @return La imagen (original) que se quiere cargar
 */
public static Imagen loadImagenGrises(String nombreImagen)
{
        Imagen ima = loadImagen(nombreImagen);
        if (ima==null)
        {
                ima=loadImagenGrises(0);//Si no existe la que han pedido, se devuelve la primera

        }
        ima.setTitulo(nombreImagen); /* se pone título a la imagen */
        return ima;
}

/**
 * Devuelve la imagen (original) que hay en la posición que se introduce
 * como parámetro
 *
 * @param numeroImagen La posición del array donde se encuentra la imagen que se
 * quiere cargar
 * @return La imagen (original) que se quiere cargar
 */
public static Imagen loadImagenGrises(int numeroImagen)
{
        //Se cargan todos los nombres
        String [] names = loadNamesImgGrises();
        //Se obtiene el nombre de la imagen que ocupa la posicion indicada
        return loadImagenGrises(names[numeroImagen]);
}


/**
 * Devuelve la imagen (en pequeño) cuyo nombre se introduce como parámetro
 * @param nombreImagen Nombre de la imagen en miniatura (sin incluir la extensión)
 * que se quiere cargar
 * @return La imagen (en pequeño) que se quiere cargar
 */
public static Image loadImagenGrisesPeque(String nombreImagen)
{
        return loadImagen(nombreImagen + MIN_IMG_GRISES).creaImage();
}
```

```
/**
 * Carga la imagen de la base de imágenes cuyo nombre se le da como
 * parámetro. Es el método que contiene la funcionalidad real de
 * la carga de imágenes en escala de grises.
 *
 * @param name Nombre de la imagen
 * @return La imagen que obtenemos
 */
private static Imagen loadImagen (String nombreImagen)
{
	Imagen imagen = null;

	Image image=null;
	try
	{
		//Esto es para cuando es un applet
		if (IMAGineApplet.applet!=null)
		{
			//Obtenemos la ruta de las imagenes (esta en una carpeta imagenes donde se alojen los .class)
			URL url = new URL (IMAGineApplet.applet.getCodeBase()+
					DIR_IMAGENESGRISES_APPLET + nombreImagen + IMAGE_EXT);
			//URL url = getClass().getResource(tituloImagen);
			image = Toolkit.getDefaultToolkit().getImage(url);
		}
		else
		{
			image = Toolkit.getDefaultToolkit().getImage(DIR_IMAGENESGRISES+
					nombreImagen + IMAGE_EXT);
		}
	}
	catch (Exception e)
	{
		e.printStackTrace();
		return null;
	}
	// Se utilzia un objeto MediaTracker para bloquear la tarea hasta
	// que la imagen se haya cargado o hayan transcurrido 10 segundos
	// desde que se inicia la carga
	MediaTracker tracker = new MediaTracker(new Button());
	tracker.addImage(image, 1);
	try
	{
		if (!tracker.waitForID(1, 10000))
		{
			System.out.println("CargaImagen: Error en la carga de la imagen");
			return null;
		}
	}
	catch (InterruptedException e)
	{
		System.out.println(e);
		return null;
	}

	imagen = new Imagen (image);

	return imagen;
}


/* *****************************************************************
 *          METODOS PARA IMAGENES A COLOR
 * ***************************************************************** */

/**
 * Devuelve los nombres de las imagenes a color(sin incluir la extensión)
 * que están en la base de imágenes del servidor
 *
 * @return array con los nombres de las imagenes a color que hay en el
 * servidor
 */
public static String [] loadNamesImgColor ()
{
	Vector vNames = new Vector();
	BufferedReader br=null;

	try
	{
		//Esto es para cuando es un applet
		if (IMAGineApplet.applet!=null)
		{
```

```
                        //Obtenemos la ruta de las imagenes (esta en una carpeta imagenes donde se alojen los .class)
                        URL rutaListado= new URL (IMAGineApplet.applet.getCodeBase()+
                                                DIR_IMAGENESCOLOR_APPLET+IMAGE_NAMESCOLOR);
                        br =new BufferedReader(new InputStreamReader(rutaListado.openStream()));
                }
                //Esto es para cuando estamos en el Editor (no es un applet) y coge el directorio distinto
                else
                {
                        br = new BufferedReader(new FileReader(DIR_IMAGENESCOLOR+IMAGE_NAMESCOLOR));
                }

                String linea="";
                while ((linea=br.readLine())!=null)
                {
                        vNames.add(linea);
                }
        }
        catch (Exception e)
        {
                e.printStackTrace();
        }
        finally
        {
                try
                {
                        if (br != null)
                                br.close(); // cierra la corriente de entrada de datos
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }
        }

        // Carga los nombres de las imágenes
        String namesImgColor[] = new String [vNames.size()];
        for (int i=0; i<namesImgColor.length;i++)
        {
                namesImgColor[i]=(String)vNames.get(i);
        }
        return namesImgColor;
}

/**
 * Método que crea un objeto ImagenColor a partir del nombre de la imagen.
 *
 * @param nombreImagen de la imagen que queremos obtener (sin extension)
 * @return ImagenColor objeto que contiene la imagen cargada.
 */
public static ImagenColor loadImagenColor (String nombreImagen)
{
        Image image=loadImage (nombreImagen);
        if (image==null)
        {
                //Retornamos la primera imagen porque se supone que al menos
                //debe haber una imagen
                return loadImagenColor(0);
        }
        else
        {
                ImagenColor imagenColor = new ImagenColor(image);
                //Ponemos el título a la imagen
                imagenColor.setTitulo(nombreImagen);
                return imagenColor;
        }
}

/**
 * Método que crea un objeto ImagenColor a partir de su posición
 * en la base de imágenes del servidor.
 *
 * @param numeroImagen con la posición de la imagen en la base de imágenes
 * del servidor.
 * @return ImagenColor objeto que contiene la imagen cargada.
 */
public static ImagenColor loadImagenColor (int numeroImagen)
{
        String [] names = loadNamesImgColor();
        return loadImagenColor(names[numeroImagen]);
}
```

```
	/**
	 * Devuelve la imagen a color (en tamaño pequeño) cuyo
	 * nombre se introduce como parámetro
	 *
	 * @param nombreImagen Nombre de la imagen a color en miniatura (sin
	 * incluir la extensión)
	 * @return El objeto Image con la imagen (en pequeño) que se quiere cargar
	 */
	public static Image loadImagenColorPeque(String nombreImagen)
	{
		return loadImage(nombreImagen + MIN_IMG_COLOR);
	}


	/**
	 * Carga una imagen a color a partir del nombre de la imagen.
	 * Es el método que contiene la funcionalidad real de la carga de
	 * imágenes a color.
	 *
	 * @param nombreImagen (nombre de la imagen sin incluir la extensión)
	 * @return objeto Image que contiene la imagen cuyo nombre hemos
	 * pasado por parámetros al método
	 */
	private static Image loadImage (String nombreImagen)
	{
		Image image=null;
		try
		{
			//Esto es para cuando es un applet
			if (IMAGineApplet.applet!=null)
			{
				//Obtenemos la ruta de las imagenes (esta en una carpeta imagenes donde se alojen los .class)
				URL url = new URL (IMAGineApplet.applet.getCodeBase() +
					DIR_IMAGENESCOLOR_APPLET+ nombreImagen + EXT_IMG_COLOR);
				image = Toolkit.getDefaultToolkit().getImage(url);
			}
			else
			{
				image = Toolkit.getDefaultToolkit().getImage(DIR_IMAGENESCOLOR +
									nombreImagen + EXT_IMG_COLOR);
			}
		}
		catch (Exception e)
		{
			e.printStackTrace();
			return null;
		}
		// Se utilzia un objeto MediaTracker para bloquear la tarea hasta
		// que la imagen se haya cargado o hayan transcurrido 10 segundos
		// desde que se inicia la carga
		MediaTracker tracker = new MediaTracker(new Button());
		tracker.addImage(image, 1);
		try
		{
			if (!tracker.waitForID(1, 10000))
			{
				System.out.println("Carga imagen: Error en la carga de la imagen");
				return null;
			}
		}
		catch (InterruptedException e)
		{
			System.out.println(e);
			return null;
		}
		return image;
	}

}
```

8.1.7. Clase Editor_BaseImgColor

```
/**
 * Aplicación java para cargar nuevas imágenes a la base de imágenes a color que se ubicará en el servidor
 * (exactamente las carga a un directorio en local que luego habrá que subir tal cual al servidor) de tal forma que
 * luego puedan ser usadas por los applets de la asignatura.
 * En el caso de imágenes a color, la imagen se guarda tal cual  (sin ningun tipo de procesado) en el directorio,
 * pero se crea también una imagen en miniatura (para mostrarla en
 * el panel de carga de imágenes ya que generarla dinámicamente en el
 * servidor ralentiza mucho el proceso) que también se guarda en dicho directorio.
 * El nombre de las imágenes que hay en el directorio figura en una fichero .txt junto a ellas.
 * Otra opción que proporciona la apliación es la de ver las imágenes que ya están en la base de imágenes del servidor y
 * la de borrar alguna imagen.
 *
 * @author Ana Belén Alonso Martín.
 * @version IMAGine 2.0 - 2008
 */
public class Editor_BaseImgColor extends Frame
implements BaseImg_Ctes, CargaImagenesCtes, ActionListener
{
    private static final long serialVersionUID = 1L;

    /**
     * textUrl es el campo de texto donde se escribe la URL desde donde se quiere cargar
     * la imagen
     *
     * textName es el campo de texto donde se escribe el nombre de la imagen que se
     * va a guardar en la base de imágenes del servidor
     */
    private TextField textUrl, textName;

    /**
     * Panel donde se podrán ver las imágenes que están en la base de imágenes del
     * servidor
     */
    private BaseImg_Scroll scrollImagenes;

    /**
     * Panel que incluye el panel donde se ven todas las imágenes de la base e
     * incluye un boton de eliminar para borrar las seleccionadas
     */
    private BaseImg_DeletePanel pDelete;

    /**
     * Panel donde el sistema pide al usuario que cargue una imagen del ordenador
     */
    private FileDialog fd;

    /**
     * Contenedor de imágenes donde se podrá visualizar la imagen que se haya cargado
     * desde el ordenador o desde una URL
     */
    private ImagenColorCanvas imc;

    /**
     * Imagen que se carga desde el ordenador o desde una URL y que se podrá
     * añadir a la base de imágenes del servidor
     */
    private ImagenColor imagenColor;

    /**
     * Frame que mostrará ventanas emergentes
     */
    private Frame fr;

    /**
     * Constructor del editor de la base de imágenes del servidor.
     * Crea el frame de la aplicación y todos los elementos gráficos
     * que lo componen
     */
    public Editor_BaseImgColor ()
    {
        super(FRAME_TITLE_COLOR);

        // se crea el panel que pedirá al usuario desde donde cargar la imagen
        fd = new FileDialog(this, CHOSE_IMAGE, FileDialog.LOAD);
        fd.setDirectory(DIRECTORIO_PORDEFECTO);
```

```
        /* Se crea la apariencia gráfica de la aplicación */
        // se ajusta el diseño de la ventana según el patrón GridBagLayout (Ver API de Java)
        setLayout(new GridBagLayout());
        Panel pfind = new Panel (new GridBagLayout());
        makeCell(pfind, new Label(URL+":"), 0, 0, 1, 1, 0.0, 0.0);
        // se crea el campo de texto de la URL
        textUrl = new TextField(40);
        makeCell(pfind, textUrl, 1, 0, 1, 1, 1.0, 0.0);
        // se crea el botón de cargar la imagen desde el ordenador
        Button bFile = new Button(FILE);
        bFile.addActionListener(this);
        makeCell(pfind, bFile, 2, 1, 1, 1, 0.0, 0.0);
        // se crea el botón de cargar la imagen desde una URL
        Button bLoad = new Button(LOAD);
        bLoad.addActionListener(this);
        makeCell(pfind, bLoad, 2, 0, 1, 1, 0.0, 0.0);
        makeCell(this, pfind, 0, 0, 1, 1, 0.0, 0.0);

        Panel pImg = new Panel(new GridBagLayout());
        // se crea el contenedor de la imagen de previsualización
        imc = new ImagenColorCanvas(null,this);
        makeCell(pImg, imc, 0, 0, 6, 6, 0.0, 0.0);
        makeCell(pImg, new Label(NEW_NAME+":"), 6, 3, 1, 1, 0.0, 0.0);
        // se crea el campo de texto del nombre de la imagen
        textName = new TextField(15);
        makeCell(pImg, textName, 7, 3, 1, 1, 0.0, 0.0);
        // se crea el botón que añade la imagen a la base de imágenes del servidor
        Button bAdd = new Button(ADD_IMAGE);
        bAdd.addActionListener(this);
        makeCell(pImg, bAdd, 7, 4, 1, 1, 0.0, 0.0);
        makeCell(this, pImg, 0, 1, 1, 1, 0.0, 0.0);
        Panel pMuestraBaseImg = new Panel(new GridBagLayout());
        // se crea el botón que muestra el estado de la base de imágenes del servidor
        Button bMuestraBaseImg = new Button(BASE_STATUS);
        bMuestraBaseImg.addActionListener(this);
        // se crea el panel que mostrará las imágenes de la base de imágenes del servidor
        scrollImagenes = new BaseImg_Scroll(0, 400, true, this);
        makeCell(pMuestraBaseImg, bMuestraBaseImg, 0, 1, 1, 1, 0.0, 0.0);
        makeCell(this, pMuestraBaseImg, 0, 2, 1, 1, 0.0, 0.0);
        pack();
        show();
}

/**
 * Ajusta el panel según el patrón GridBagLayout
 */
private void makeCell(Container cont, Object arg, int x, int y, int w, int h, double weightx, double weighty)
{
        GridBagLayout gbl = (GridBagLayout)cont.getLayout();
        GridBagConstraints c = new GridBagConstraints();
        Component comp;
        c.fill = GridBagConstraints.NONE;
        c.gridx = x;
        c.gridy = y;
        c.gridwidth = w;
        c.gridheight = h;
        c.weightx = weightx;
        c.weighty = weighty;
        c.insets = new Insets(5,5,5,5);
        comp=(Component)arg;
        cont.add(comp);
        gbl.setConstraints(comp,c);
}

/**
 * Escala una image a las dimensiones que se pasan
 * por parámetros al métodos
 * @param im La imagen que queremos escalar
 * @param anchoNew ancho que queremos que tenga la imagen reescalada
 * @param altoNew alto que queremos que tenga la imagen reescalada
 * @return La imagen escalada
 */
private Image reescala (Image im, int anchoNew, int altoNew)
{
        if(im==null)
        {
                return null;
        }
        // se devuelve una imagen reescalada por las nuevas dimensiones
        return im.getScaledInstance(anchoNew,altoNew,Image.SCALE_SMOOTH);
}
```

```
/**
 * Carga una imagen en el contenedor a partir de la URL que aparece
 * escrita en el campo de texto para tal fin en la aplicación.
 * La imagen la reescala para que tenga unas dimensiones determinadas
 * al mostrarse en el contenedor del frame.
 */
private void cargaImagenDesdeURL()
{
	try
	{
		// se obtiene la Image a partir de la url (la obtenemos del campo de texto)
		String ruta = textUrl.getText().trim();
		Image image = getToolkit().getImage(new URL(ruta));

		// Se utiliza un objeto MediaTracker para bloquear la tarea hasta
		// que la imagen se haya cargado o hayan transcurrido 10 segundos
		// desde que se inicia la carga
		MediaTracker tracker = new MediaTracker(new Button());
		tracker.addImage(image, 1);
		try
		{
			if (!tracker.waitForID(1, 10000))
			{
				System.out.println("Error en la carga de la imagen");
				System.exit(1);
			}
		}
		catch (InterruptedException e)
		{
			System.out.println(e);
		}

	// Creamos un objeto ImagenColor a partir de la Image obtenida
		//(esto será lo que se guardará en la BBII
	imagenColor = new ImagenColor(image);

		// Reescalamos la image al tamaño del contenedor para mostrarla en el contenedor
	// de la aplicación (aunque en la base de imágenes, en el caso de imágenes a color,
	// se guardan con su tamaño original.
	Image imageReescalada = reescala (image, ANCHO_IMG_BASE, ALTO_IMG_BASE);
	// Creamos un objeto ImagenColor a partir de la Image reescalada para mostrarla en el contenedor
	ImagenColor imagenColorReescalada = new ImagenColor(imageReescalada);
		// se le pone el título a la imagen cargada
	String nombreImagen = ruta.substring(Math.max(ruta.lastIndexOf("/"),ruta.lastIndexOf("\\"))+1,ruta.lastIndexOf("."));
		imagenColor.setTitulo(nombreImagen);
		imagenColorReescalada.setTitulo(nombreImagen);
		//Se muestra el nombre de la imagen.
		textName.setText(nombreImagen);
	// Pinta en el contenedor la imagen reescalada
	imc.actualiza(imagenColorReescalada);
	}
	catch(Exception e)
	{
		JOptionPane.showMessageDialog(this, "Error al cargar la imagen de la URL" , "Mensaje de error",
						JOptionPane.ERROR_MESSAGE);
		System.err.println(e.getMessage());
		// si no se encuentra ninguna imagen en esa URL no carga nada
		textName.setText("");
		imc.actualiza(null);
		imagenColor = null;
	}
}

/**
 * Borra de la base de imágenes del servidor las imagenes
 * seleccionadas en el panel estado actual base de imágenes
 */
private void borraImagenes()
{
   //Recoge los nombres de las imágenes que están seleccionadas
   String namesSelected[] = scrollImagenes.getNombresImagenesSeleccionadas();
   //Recoge los nombres de todas las imágenes
   String names[] = scrollImagenes.getNames();
   int k=0;

   //Almacena en un array aquellos nombres que no van a ser borrados
   String newnames[] = new String[names.length-namesSelected.length];
   for(int i=0;i<names.length;i++)
   {
      for(int j=0;j<namesSelected.length;j++)
      {
```

```
                    if(namesSelected[j].equals(names[i]))
                    {
                            break;
                    }

                    if (j==namesSelected.length-1)
                    {
                            newnames[k++]=names[i];
                    }
            }
        }
        // Guarda los nombres obtenidos en el archivo "listadoImagenesColor.txt"
        saveNames(newnames);
        scrollImagenes.update();

            // Borramos los ficheros correspondientes a esta imagen del servidor
            File file;
            for(int m=0; m<namesSelected.length;m++)
            {
                    try
                    {
                            String base = DIR_IMAGENESCOLOR+namesSelected[m];
                            // borra los dos tipos de archivos que tiene la imagen
                            file = new File(base+EXT_IMG_COLOR);
                            file.delete();
                            file = new File(base+MIN_IMG_COLOR+EXT_IMG_COLOR);
                            file.delete();
                    }
                    catch (Exception e)
                    {
                            System.out.println ("No se borra la imagen porque ya no existe");
                    }
            }

            JOptionPane.showMessageDialog(this, "Imagen(es) borrada(s) correctamente.",
                                            "Proceso Borrado Imágenes",JOptionPane.INFORMATION_MESSAGE);
    }


    /**
     * Graba en la base de imágenes del servidor
     * (realmente graba la imagen en el directorio
     * en local que luego subiremos al servidor)
     * la imagen cargada en el Editor
     */
    private void grabaImagenes()
    {
            //Se obtiene el nombre de la imagen a grabar del campo de texto correspondiente
            String name = textName.getText();

            /* Se guarda la imagen en el directorio en local que luego
             * subiremos al servidor y constituirá la base de imágenes a color */
            saveImage(imagenColor.getBufferedImagen(), name + EXT_IMG_COLOR);

            /* Se guarda también la imagen en miniatura */

            // Se crea la imagen en pequeño para el panel de selección de la base de imágenes
            // (Esta imagen se guarda también en el directorio porque generarla dinámicamente
            // desde el servidor a partir de la imagen de dimensiones normales es muy lento)
            // (esto, hecho en local, es rápido y apenas costoso, pero al ejecutarlo en el servidor,
            // la generación dinámica de las imágenes reescaladas para mostrarlas en el panel
            // de carga resultaba muy lento).
            Image imagePeque = reescala(imagenColor.getImage(), ANCHO_IMGCOLOR_CARGA,
ALTO_IMGCOLOR_CARGA);
            // guarda la imagen en pequeño
            saveImage(new ImagenColor(imagePeque).getBufferedImagen(), name + MIN_IMG_COLOR + EXT_IMG_COLOR);


            /* Se actualiza el fichero que contiene el nombre de las imágenes guardadas */

            // Se obtienen los nombres de las imagenes que había en la base de imágenes y se añade la nueva
            String [] names = CargaImagenes.loadNamesImgColor();
            String [] newnames;
            if (names!=null && names.length>0)
            {
                    newnames = new String [names.length+1];
                    for (int i=0;i<names.length;i++){
                            newnames [i]=names[i];
                    }
                    newnames[names.length]=name;
            }
```

```
        else
        {
                newnames = new String [1];
                newnames [0]= name;
        }

        // Se guarda en el fichero los nombres de todas las imagenes almacenadas en la BBII
        saveNames(newnames);
        // Se actualiza el panel que muestra las imagenes almacenadas.
        scrollImagenes.update();
        String rutaSaveImage=DIR_IMAGENESCOLOR+name;
        JOptionPane.showMessageDialog(this, "Imagen '" + name +"' almacenada correctamente en:\n" +
                        rutaSaveImage,"Proceso Grabación Imágenes",JOptionPane.INFORMATION_MESSAGE);
    }

    /**
     * Se guarda la imagen en el directorio en local que luego,
     * al subirlo al servidor, será la base de imágenes a color.
     *
     * @param bImage Imagen que queremos guardar
     * @param nombre Título de la imagen
     */
    private void saveImage(BufferedImage bImage, String nombre)
    {
        String rutaSaveImage=DIR_IMAGENESCOLOR+nombre;
        File file = new File(rutaSaveImage);
        try
        {
            ImageIO.write(bImage,"jpg",file);
        }
        catch (IOException ioe)
        {
                JOptionPane.showMessageDialog(this, "Error al guardar la imagen." + "\nNo es posible guardar la imagen
en la ruta:" +  rutaSaveImage,"Mensaje de error",                JOptionPane.ERROR_MESSAGE);
                System.err.println(ioe.getMessage());
        }
    }

    /**
     * Guarda los nombres de las imágenes que queremos tener en
     * la base de imágenes en un fichero .txt
     *
     * @param names Títulos de las imágenes que queremos guardar
     */
    private void saveNames(String names[])
    {
        String rutaSaveListado=DIR_IMAGENESCOLOR +IMAGE_NAMESCOLOR;
        try
        {
                BufferedWriter bf = new BufferedWriter (new FileWriter (rutaSaveListado,false));
                // se crea el flujo de datos saliente
                for (int i=0;i<names.length;i++)
                {
                        bf.write(names[i]+"\n");
                }
                bf.close();
        }
        catch(Exception e)
        {
                JOptionPane.showMessageDialog(this,  "Error al guardar el nombre de la imagen" +
                                                        "\nNo es posible grabar el fichero "+rutaSaveListado ,
                                                          "Mensaje de error", JOptionPane.ERROR_MESSAGE);
        }
    }

    /**
     * Muestra el panel para que el usuario elija donde se encuentra la imagen que
     * quiere cargar en un directorio de su ordenador.
     */
    private void seleccionaFichero()
    {
        fd.show();
        String ruta="";
        if(fd.getFile() != null)
        {
                // obtiene la imagen del directorio del ordenador
                //(en realidad lo hace a partir de una URL)
                ruta = "file:/"+fd.getDirectory()+fd.getFile();
                textUrl.setText(ruta);
        }
    }
```

```
/**
 * Abrimos la ventana con el estado actual de la base de imágenes del
 * servidor y un botón de eliminar
 */
private void sacaPanelEstadoActualBaseImg()
{
	pDelete= new BaseImg_DeletePanel(scrollImagenes,this);
	sacaVentana(BASE_STATUS, pDelete, false, new Point(100,50));
}


//******** MÉTODOS AUXILIARES PARA MOSTRAR LAS VENTANAS EMERGENTES **********
/**
 * Saca una ventana emergente que muestra el panel que pasamos por parámetros
 * @param tituloVentana
 * @param panel que mostraremos en la nueva ventana
 * @param ancho de la ventana
 * @param alto de la ventana
 */
private void sacaVentana (String tituloVentana, Panel panel, int ancho, int alto)
{
	//Ventana donde se mostrará el resultado de la operación seleccionada
	fr = new Frame();
	fr.setTitle(tituloVentana);
	fr.add(panel);
	fr.setVisible(true);
	fr.setSize(ancho, alto);
	fr.validate();

	fr.addWindowListener(new WindowAdapter(){
	public void windowClosing(WindowEvent e){
			Window w = e.getWindow();
			w.removeAll();
			w.dispose();
			}
});
}


/**
 * Abre una nueva ventana mostrando el componente p
 *
 * @param tit  Título de la ventana
 * @param p    Componente que se va a mostrar en la ventana
 * @param b    Muestra si se puede cambiar el tamaño de la imagen
 * @param po   Punto donde se va a mostrar la ventana
 */
private void sacaVentana(String tit, Component p, boolean b, Point po)
{
	// se crea la ventana
	fr = new Frame(tit);
	fr.setResizable(b);
	// se añade el gestor de eventos de ventana
	fr.addWindowListener(new WindowAdapter(){
	public void windowClosing(WindowEvent e){
			Window w = e.getWindow();
			w.removeAll();
			w.dispose();
			}
	});

	fr.add(p);
	fr.pack();

	// se determina la dimensión y la posición de la ventana
	Dimension e = getSize();
	Dimension d = fr.getSize();
	int a = po.x + e.width / 2 - d.width / 2;
	int c = po.y + e.height / 2 - d.height / 2;
	fr.setLocation(a > 0 ? a : 0, c > 0 ? c : 0);
	fr.dispose(); // Bug en visualizadores jdk1.1.3
	// se muestra la ventana en pantalla
	fr.show();
}
```

```
    /**
     * Quita la ventana emergente que pudiese haber abierta
     */
    private void quitaVentana ()
    {
            if (fr!=null)
            {
                    //Se cierra la ventana donde se muestra el tipo de filtro
                    fr.removeAll();
                    fr.dispose();
                    fr = null;
            }
    }


    /**
     * Método que escucha los eventos del Editor (cualquier botón
     * que se pulse) y llama al método indicado para realizar la
     * acción adecuada.
     */
    public void actionPerformed(ActionEvent evt)
    {
        //Label del botón que ha generado el evento
        String comando=evt.getActionCommand();

        //Cargar una imagen desde fichero
        if (comando.equals (FILE))
        {
           seleccionaFichero();
           cargaImagenDesdeURL();
        }

        //Cargar una imagen desde URL
        else if (comando.equals (LOAD))
        {
           cargaImagenDesdeURL();
        }

        //Añadir imagen
        else if (comando.equals (ADD_IMAGE))
        {
           grabaImagenes();
        }

        //Botón Estado actual de la base de imágenes
        else if (comando.equals (BASE_STATUS))
        {
          sacaPanelEstadoActualBaseImg();
        }

        //Eliminar imagenes seleccionadas
        else if (comando.equals (REMOVE))
        {
           borraImagenes();
           quitaVentana();
        }
    }

    /**
     * Método principal del programa que lanza su ejecución
     *
     * @param args Parámetros de entrada del programa al cargarlos
     * (no hace falta ninguno)
     */
    public static void main(String args[])
    {
            // crea un nuevo editor
            Editor_BaseImgColor editor = new Editor_BaseImgColor();
            // Para cerrar la ventana
            editor.addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                            System.exit(0);
                            }
            });
    }

}
```

8.2. CLASES MÁS IMPORTANTES DEL CURSO RESTAURACIÓN DE IMÁGENES

8.2.1. Clase Imagen

```
/**
 * La clase <code>Imagen</code> implementa una Imagen digital. Almacena los
 * pixels que componenen la imagen, su titulo, e incluso su transformada de
 * Fourier si ha sido necesario calcularla.
 * Ademas la clase proporciona toda una serie de metodos que realizan
 * operaciones de tratamiento digital de imagenes.
 *
 * @author Realizado y documentado por Pablo Puente
 * @author Modificado por Fernando Hoyos
 * @author Modificado por Ana Belén Alonso Martín: métodos para curso
 * restauración de imágenes
 * @version IMAGine 2.0 - 2008
 */
public class Imagen implements Serializable
{
    /**
     * Identificador de la versión de la clase
     */
    private static final long serialVersionUID = 4697613170221019684L;

    /**
     * El array en el que se almacenan los pixels de la imagen, el primer indice
     * corresponde al ancho y el segundo al alto.
     */
    transient private int p[][];

    /**
     * El ancho en pixels de la imagen.
     */
    private int w;

    /**
     * El alto en pixels de la imagen.
     */
    private int h;

    /**
     * El titulo de la imagen. Puede modificarse segun se vayan realizando
     * operaciones.
     */
    private String titulo = "";

    /**
     * La transformada de Fourier de la imagen actual.
     */
    private transient FFTImagenGrises tf;

    /**
     * Número de iteraciones necesarias para realizar las operaciones
     * morfológicas.
     */
    private int niteraciones = 0;

    /**
     * Objeto de la clase ProcMorf necesario para realizar las operaciones
     * morfológicsa.
     */
    private ProcMorf pm;

    /**
     * Objeto de la clase RestauracionImg necesario para realizar las
     * operaciones del curso restauración de imagenes.
     */
    private RestauracionImg restImg;

    // ----------CONSTRUCTORES----------
    /**
     * Construye una imagen con la primera imagen de la base de Imagenes.
     */
    public Imagen()
    {
        this(0);
    }
```

```
/**
 * Construye una imagen con la i-esima imagen de la base de Imagenes.
 *
 * @param i posicion de la imagen a recuperara en la base de imagenes.
 * La primera imagen es la 0.
 */
public Imagen(int i)
{
	this(CargaImagenes.loadImagenGrises(i));
}

/**
 * Construye una imagen con imagen de la base de Imagenes que tiene el
 * título que pasamos por parámetros al método.
 *
 * @param nombreImagen titulo de la imagen que se quiere cargar.
 */
public Imagen(String nombreImagen)
{
	this(CargaImagenes.loadImagenGrises(nombreImagen));
}

/**
 * Construye una imagen como copia de otra.
 *
 * @param im imagen a copiar.
 */
public Imagen(Imagen im)
{
	if (im == null)
		return;
	w = im.w;
	h = im.h;
	titulo = im.titulo;
	int i, j;
	p = new int[w][h];
	// Como no conseguimos que funcionase la clonacion, copiamos los pixels
	// a mano
	for (i = 0; i < w; i++)
	{
		for (j = 0; j < h; j++)
		{
			p[i][j] = im.p[i][j];
		}
	}
}

/**
 * Construye una imagen vacia(negra) de dimensiones <code>wi</code> y
 * <code>he</code>.
 *
 * @param wi ancho de la imagen construida.
 * @param he alto de la imagen construida.
 */
public Imagen(int wi, int he)
{
	w = wi;
	h = he;
	p = new int[h][w];
}

/**
 * Construye una Imagen a partir de un objeto Image.
 *
 * @param image objeto Image a partir del cual construir la imagen.
 */
public Imagen(Image image)
{
	creaImagen(image);
}

/**
 * Crea una Imagen a partir de la URL <code>ucompleta</code>.
 *
 * @param ucompleta URL en la que se encuentra el fichero grafico
 * del cual descargar la imagen.
 * @param tit titulo que se quiera dar a la imagen.
 */
public Imagen(URL url, String tit)
{
	titulo = tit;
```

```
        Image in = Toolkit.getDefaultToolkit().getImage(url);
        FuncionesImage.waitForImage(in); // wait for image to load before painting
        creaImagen(in);
}

// ---------MÉTODOS AUXILIARES DE LOS CONSTRUCTORES (Ej.CONVERSIÓN 'IMAGE <=> IMAGEN')---------

/**
 * Rellena el array de pixels de la imagen actual a partir de
 * <code>image</code>.
 *
 * @param image objeto del cual leer los pixels.
 */
private void creaImagen(Image image)
{
        w = image.getWidth(null);
        h = image.getHeight(null);
        int pixels[] = new int[w * h];
        // El PixelGrabber sirve para grabar los pixels de la imagen en un array
        PixelGrabber pg = new PixelGrabber(image, 0, 0, w, h, pixels, 0, w);
        try
        {
                pg.grabPixels();
        }
        catch (InterruptedException e)
        {
                System.err.println("interrupted waiting for pixels!");
                return;
        }
        p = new int[h][w];
        int i, j;

        // Alamcenamos los pixels de la imagen conviritendolos a tonos de
        // grises, para ello se suman las tres componentes y se dividen entre 3
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        p[i][j] = ((pixels[i * w + j] & 255)
                                        + ((pixels[i * w + j] >> 8) & 255) + ((pixels[i * w + j] >> 16) & 255)) / 3;
                }
        }
        tf = null;
}

/**
 * Crea un objeto Image a partir de esta imagen.
 *
 * @return objeto Image correspondiente a esta imagen.
 */
public Image creaImage()
{
        int newpixels[] = new int[w * h];
        int i, j, a;
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        a = p[i][j];
                        // Crear el pixel RGB (en gris) correspondiente a este pixel
                        // expresado como nivel de gris.
                        newpixels[i * w + j] = (255 << 24) | (a << 16) | (a << 8) | a;
                }
        }
        Image img;
                img = Toolkit.getDefaultToolkit().createImage(new MemoryImageSource(w, h, newpixels, 0, w));
        return img;
}


// ----------------MÉTODOS GET/SET ATRIBUTOS CLASE----------------------

/**
 * Devuelve el titulo de la imagen.
 *
 * @return el titulo de la imagen.
 */
public String getTitulo()
{
        return titulo;
}
```

```
/**
 * Cambia el titulo de la imagen.
 * @param nuevoTitulo el nuevo titulo que se quiere poner a la imagen.
 */
public void setTitulo(String nuevoTitulo)
{
	titulo = nuevoTitulo;
}

/**
 * Devuelve un array de enteros bidimensional que representa la imagen. Los
 * valores de los enteros se encuentran entre 0 y 255.
 * @return el array bidimensional de enteros de los pixels.
 */
public int[][] getPixels()
{
	return p;
}

/**
 * Modifica los pixels de la imagen. El usuario de la clase debe encargarse
 * de mantenr los valores del ancho y alto sincronizados con el array de
 * pixels.
 *
 * @param newPixels Array bidimensional de enteros comprendidos entre
 * 0 y 255 que representan la imagen.
 */
public void setPixels(int newPixels[][])
{
	p = newPixels;
}

/**
 * Devuelve el ancho en pixels de la imagen.
 * @return el ancho en pixels de la imagen.
 */
public int getWidth()
{
	return w;
}

/**
 * Cambia el ancho en pixels de la imagen.
 * @param newWidth nuevo ancho en pixels de la imagen.
 */
public void setWidth(int newWidth)
{
	w = newWidth;
}

/**
 * Devuelve el alto en pixels de la imagen.
 * @return el alto en pixels de la imagen.
 */
public int getHeight()
{
	return h;
}

/**
 * Cambia el alto en pixels de la imagen.
 * @param newHeight nuevo alto en pixels de la imagen.
 */
public void setHeight(int newHeight)
{
	h = newHeight;
}

/**
 * Devuelve el numero de iteraciones necesarias para realizar una
 * operación..
 * @return el número de iteraciones.
 */
public int getIteraciones()
{
	return niteraciones;
}
```

```
// ----------MÉTODOS AUXILIARES----------

/**
 * Modifica la imagen actual, convirtiendola en cuadrada de dimension
 * <code>size</code>, rellenando el espacio sobrante de negro.
 *
 * @param size tamaño de la nueva imagen cuadrada
 */
public void centra(int size)
{
	int nu[][] = new int[size][size];
	int i, j;
	for (i = 0; i < h; i++)
	{
		for (j = 0; j < w; j++)
		{
			nu[(size - h) / 2 + i][(size - w) / 2 + j] = p[i][j];
		}
	}

	p = nu;
	w = h = size;
	tf = null;
}

/**
 * Corta los valores negativos a 0 y rescala los positivas hasta 255.
 */
public void cropscale()
{
	int i, j, max = -256;
	for (i = 0; i < h; i++)
	{
		for (j = 0; j < w; j++)
		{
			if (p[i][j] > max)
				max = p[i][j];
		}
	}
	for (i = 0; i < h; i++)
	{
		for (j = 0; j < w; j++)
		{
			if (p[i][j] < 0)
				p[i][j] = 0;
			else
				p[i][j] = 255 * p[i][j] / max;
		}
	}
	tf = null;
}

/**
 * Recompone los 4 cuadrantes de una imagen que representa el modulo o la
 * fase de la transformada de Fourier de otra para que los ejes aparezcan
 * centrados en la imagen.
 */
public void fftshift()
{
	int i, j, t;
	for (i = 0; i < h / 2; i++)
	{
		for (j = 0; j < w / 2; j++)
		{
			t = p[i][j];
			p[i][j] = p[i + h / 2][j + w / 2];
			p[i + h / 2][j + w / 2] = t;
		}
		for (j = w / 2; j < w; j++)
		{
			t = p[i][j];
			p[i][j] = p[i + h / 2][j - w / 2];
			p[i + h / 2][j - w / 2] = t;
		}
	}
}
```

```
// ----------MÉTODOS DE LAS OPCIONES DEL SUBMENÚ----------

/**
 * Crea un objeto Image a parir de una zona de esta imagen aumentandola un
 * determinado <code>factor</code>.
 *
 * @param x coordenada x del centro de la zona de aumento.
 * @param y coordenada y del centro de la zona de aumento.
 * @param ancho ancho en pixels de la Image creada.
 * @param alto alto en pixels de la Image creada.
 * @param factor factor de aumento.
 * 1 pixel en la imagen original=<code>factor</code> pixels en la aumentada.
 * @return la Image conteniendo el zoom de esta imagen.
 */
public Image zoomImage(int x, int y, int ancho, int alto, int factor)
{
	int newpixels[] = new int[ancho * alto];
	int i, j, a;

	for (i = 0; i < alto; i++)
	{
		for (j = 0; j < ancho; j++)
		{
			a = p[(y + (i - alto / 2) / factor + h) % h][(x
					+ (j - ancho / 2) / factor + w)	% w];
			// Crear el pixel RGB (en gris) correspondiente a este pixel
			// expresado como nivel de gris.
			newpixels[i * ancho + j] = (255 << 24) | (a << 16) | (a << 8)| a;
		}
	}
	Image img = Toolkit.getDefaultToolkit().createImage(new MemoryImageSource(ancho, alto, newpixels, 0, ancho));
	return img;
}

/**
 * Devuelve la transformada de Fourier de la imagen. Si es necesario se
 * realiza el calculo de esta si no se dispone de ella.
 *
 * @return transformada de Fourier de la imagen.
 */
public FFTImagenGrises getTf()
{
	if (tf == null)
		tf = new FFTImagenGrises(this);
	return tf;
}

/**
 * Cambia la transformada de Fourier de la imagen. En principio, el usuario
 * no debria utilizar este metodo, se usa por ejemplo Para indicar la
 * transformada de Fourier ya conocida de imagenes que provienen del dominio
 * de la frecuencia.
 *
 * @param newTf nuevaTransformada de Fourier.
 */
public void setTf(FFTImagenGrises newTf)
{
	tf = newTf;
}


/**
 * Invierte los tonos de la imagen.
 */
public void invierte()
{
	int i, j;
	for (i = 0; i < h; i++)
		for (j = 0; j < w; j++)
			p[i][j] = 255 - p[i][j];
	tf = null;
}

/**
 * Realiza la ecualizacion del histograma de la imagen para aumentar el
 * contraste de esta.
 */
public void ecualizaHistograma()
{
	int cum[] = new int[256], hist[] = new int[256], cnt = 0, sofar = 0, x, i, j;
```

```
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        cum[p[i][j]]++; // count number of values
                        cnt++; // count total number in range
                }
        }

        for (x = 0; x < 256; x++)
        {
                sofar += cum[x];
                hist[x] = sofar * 255 / cnt;
        }
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        p[i][j] = hist[p[i][j]]; // return cumulative distribution
                }
        }
        tf = null;
}

/**
 * Calcula el histograma de la imagen.
 *
 * @return un array de int de dimension 256 con el numero de pixels
 * que hay en la imagen de cada tono.
 */
public int[] histograma()
{
        int i, j, cum[] = new int[256];
        // for (int x=0; x<256; x++) cum[x] = 0; No es necesario, en Java los
        // ints se inicizalizan a 0
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        cum[p[i][j]]++; // count number of values
                }
        }
        return cum;
}

public void gira(float radianes)
{
        gira(radianes, false);
}

/**
 * Gira la imagen utilizando el metodo del punto mas cercano o el de
 * interpolacion lineal. Los puntos que se salen del area de la imagen
 * aparden por el otro extremo de la imagen. Extraido de "Image Processing,
 * Analysis, and Machine Vision" Milan Sonka. Pag 66.
 *
 * @param radianes
 * @param mas_cercano
 *          <code>true</code> se utiliza el metodo del mas cercano.;
 *          <code>false</code> se utiliza el metodo de interpolacion
 *          lineal.
 */
public void gira(float radianes, boolean mas_cercano)
{
        int i, j, l, k;
        int nu[][] = new int[h][w];
        double co, si, a, b, x, y;
        for (i = 0; i < w; i++)
        {
                for (j = 0; j < h; j++)
                {
                        co = Math.cos(radianes);
                        si = Math.sin(radianes);
                        x = (i - w / 2) * co + (j - h / 2) * si + w / 2;
                        y = -(i - w / 2) * si + (j - h / 2) * co + h / 2;
                        l = (int) Math.round(x);
                        k = (int) Math.round(y);
                        if (mas_cercano)
                        {
                                a = b = 0;
                        }
```

```
                                else
                                {
                                        a = x - l;
                                        b = y - k;
                                }
                                nu[i][j] = (int) ((1 - a) * (1 - b)
                                                * p[(l + 4 * w) % w][(k + 4 * h) % h] + a * (1 - b)
                                                * p[(l + 1 + 4 * w) % w][(k + 4 * h) % h] + b * (1 - a)
                                                * p[(l + 4 * w) % w][(k + 1 + 4 * h) % h] + a * b
                                                * p[(l + 1 + 4 * w) % w][(k + 1 + 4 * h) % h]);
                                // Si al redondear algun valor se sale de madre, lo corregimos
                                if (nu[i][j] > 255)
                                        nu[i][j] = 255;
                                if (nu[i][j] < 0)
                                        nu[i][j] = 0;
                        }
        }
        p = nu;
        titulo = "Girada de " + titulo;
        tf = null;
}

/**
 * Realiza la convolucion de la mascara con la imagen. Despues del filtrado
 * realizad un postprocesado u otro en funcion del tipo de mascara.
 *
 * @param m mascara con la que realizar la convolucion
 */
public void filtra(Mascara m)
{
        int s, i, j, k, l;
        int fp[][] = new int[h][w];
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        s = 0;
                        for (k = 0; k < m.dim; k++)
                        {
                                for (l = 0; l < m.dim; l++)
                                {
                                        s += p[(i + k) % h][(j + l) % w] * m.masc[k][l];
                                }
                        }
                        fp[(i + m.dim / 2) % h][(j + m.dim / 2) % w] = s / m.divisor;
                }
        }
        p = fp;
        switch (m.post)
        {
                case 1:
                        margenCorta();
                        break;
                case 2:
                        margenDinamico();
                        break;
                case 3:
                        ecualizaHistograma();
                        break;
                case 4:
                        cortaCero();
                        break;
        }
        titulo = "Filtrado máscara de " + titulo;
        tf = null;
}

/**
 * Método auxiliar caso 1 de filtra(): Fuerza que los pixels con valores
 * menores de 0 valgan 0 y los mayores de 255 valgan 255
 */
public void margenCorta()
{
        int i, j;
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        p[i][j] = p[i][j] >= 0 ? p[i][j] : 0;
                        p[i][j] = p[i][j] < 256 ? p[i][j] : 255;
                }
```

```
        }
        tf = null;
}

/**
 * Método auxiliar caso 2 de filtra(): Reescala el rango de grises de la
 * imagen para que coincida con el rango dinamico completo, es decir para
 * que los tonos entren entre 0 y 255.
 */
public void margenDinamico()
{
        int i, j, min = Integer.MAX_VALUE, max = Integer.MIN_VALUE;
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        if (p[i][j] < min)
                        {
                                min = p[i][j];
                        }
                        if (p[i][j] > max)
                        {
                                max = p[i][j];
                        }
                }
        }
        if (min == max)
        {
                return;
        }
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        p[i][j] = (255 * (p[i][j] - min)) / (max - min);
                }
        }
        tf = null;
}

/**
 * Método auxiliar caso 4 de filtra(): Colorea de blanco los valores
 * cercanos a cero (valor absoluto menor que 10) y de negro el resto
 */
public void cortaCero()
{
        int i, j;
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        p[i][j] = (Math.abs(p[i][j]) < 10) ? 255 : 0;
                }
        }
        tf = null;
}

/**
 * Realiza un filtrado de mediana de tamaño <code>tam</code>.
 *
 * @param tam Tamaño de la "mascara" de filtrado que define
 * la cantidad de puntos vecinos utilizados
 */
public void filtraMediana(int tam)
{
        int i, j, k, l;
        int v[] = new int[tam * tam];
        int fp[][] = new int[h][w];
        for (i = 0; i < h; i++)
        {
                for (j = 0; j < w; j++)
                {
                        for (k = 0; k < tam; k++)
                        {
                                for (l = 0; l < tam; l++)
                                {
                                        v[k + l * tam] = p[(i + k) % h][(j + l) % w];
                                }
                        }
                        //Aplicamos el algoritmo QSortAlgoritm
                        FiltradoMedianaAlgoritmo.aplicaQSortAlgorithm(v);
```

```
                        fp[(i + tam / 2) % h][(j + tam / 2) % w] = v[(tam * tam + 1) / 2];
                }
        }
        p = fp;
        titulo = "Filtrado mediana de " + titulo;
        tf = null;
}

/**
 * Añade ruido gausiano de media <code>media</code> y varianza
 * <code>varianza</code> a la imagen.
 *
 * @param media media del ruido gausiano
 * @param varianza varianza del ruido gausiano
 */
public void ruidoGaussiano(float media, float varianza)
{
        Random r = new Random();
        int nuevo, i, j;
        for (i = 0; i < w; i++)
        {
                for (j = 0; j < h; j++)
                {
                        nuevo = (int) ((float) p[i][j] + media +
                                        (float) Math.sqrt(varianza) * r.nextGaussian());
                        // En caso de que el nuevo valor se salga del rango, lo
                        // descartamos
                        if ((nuevo >= 0) && (nuevo < 256))
                        {
                                p[i][j] = nuevo;
                        }
                }
        }
        titulo = titulo + " con Ruido Gaussiano";
        tf = null;
}

/**
 * Añade ruido impulsivo con un cierto porcentaje que se introduce como
 * parámetro. El ruido impulsivo consiste en que para cada pixel se crea un
 * número aleatorio (entre 0 y 1) y se comprueba con la frecuencia
 * seleccionada (probabilidad de que a ese pixel se le añada ruido). Si ese
 * número es menor que la frecuencia, la mitad de los casos ese pixel será
 * blanco (valor 0) y la otra mitad el pixel será negro (valor 255)
 *
 * @param porcentaje Frecuencia del ruido impulsivo
 */
public void ruidoImpulsivo(float porcentaje)
{
        Random r = new Random();
        int i, j;
        float f;
        for (i = 0; i < w; i++)
        {
                for (j = 0; j < h; j++)
                {
                        // Obtenemos un nuevo número aleatorio (entre 0 y 1)
                        f = r.nextFloat();
                        if (f < porcentaje / 2)
                        {
                                p[i][j] = 0;
                        }
                        else if ((f >= porcentaje / 2) && (f < porcentaje))
                        {
                                p[i][j] = 255;
                        }
                }
        }
        titulo = titulo + " con Ruido Impulsivo";
        tf = null;
}
```

```
/**
 * Aplica una funcion de trasferencia de tonos a la imagen.
 *
 * @param func La funcion de trasnferencia a aplicar.
 */
public void transferencia(Transferencia func)
{
	int trans[] = new int[256], i, j;
	for (i = 0; i < 256; i++)
	{
		trans[i] = func.evalua(i);
	}
	for (i = 0; i < w; i++)
	{
		for (j = 0; j < h; j++)
		{
			p[i][j] = trans[p[i][j]];
		}
	}
	titulo = "Procesada por punto de " + titulo;
	tf = null;
}

// ---------MÉTODOS DEL CURSO PROCESADO MORFOLÓGICO----------
/**
 * Inicializamos el objeto pm pasándole la imagen que existe en el momento.
 */
public void inicializaProcMorf()
{
	pm = new ProcMorf(this);
}

/**
 * Devuelve el atributo pm.
 *
 * @return el atributo pm.
 */
public ProcMorf getProcMorf()
{
	return pm;
}

/**
 * Aplica una operación morfológica binaria
 *
 * @param e EE con el que vamos a realizar la operación
 * @param operacion String que especifica la operación a realizar
 */
public void operaMorf(EE e, String operacion)
{
	inicializaProcMorf();
	if (operacion.equals("Dilatación"))
	{
		p = pm.dilatacion(e);
		titulo = "Dilatación de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Erosión"))
	{
		p = pm.erosion(e);
		titulo = "Erosión de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Apertura"))
	{
		p = pm.apertura(e);
		titulo = "Apertura de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Cierre"))
	{
		p = pm.cierre(e);
		titulo = "Cierre de " + titulo;
		tf = null;
	}
}
```

```
/**
 * Aplica el algoritmo HitMiss
 * @param e EE con el que vamos a realizar la operación
 */
public void operaHitMiss(EE e)
{
	inicializaProcMorf();
	p = pm.hitMiss(e);
	titulo = "HitMiss de " + titulo;
	tf = null;
}

/**
 * Aplica un algoritmo morfológico
 * @param e EE con el que vamos a realizar la operación
 * @param borde String que especifica el algoritmo a realizar
 */
public void bordes(EE e, String borde)
{
	inicializaProcMorf();
	int[][] dst1 = new int[h][w];
	int[][] dst2 = new int[h][w];
	if (borde.equals("Borde Exterior"))
	{
		dst1 = p;
		p = pm.dilatacion(e);
		p = pm.resta(p, dst1);
		titulo = "Borde Exterior de " + titulo;
		tf = null;
	}
	else if (borde.equals("Borde Interior"))
	{
		dst1 = p;
		p = pm.erosion(e);
		p = pm.resta(dst1, p);
		titulo = "Borde Interior de " + titulo;
		tf = null;
	}
	else if (borde.equals("Borde Ancho"))
	{
		dst2 = pm.dilatacion(e);
		inicializaProcMorf();
		dst1 = pm.erosion(e);
		p = pm.resta(dst2, dst1);
		titulo = "Borde Ancho de " + titulo;
		tf = null;
	}
	else if (borde.equals("Adelgazar"))
	{
		p = pm.adelgazamientoTotal(e);
		niteraciones = pm.niteraciones;
		titulo = "Adelgazamiento de " + titulo;
		tf = null;
	}
	else if (borde.equals("Engordar"))
	{
		p = pm.engordeTotal(e);
		niteraciones = pm.niteraciones;
		titulo = "Engorde de " + titulo;
		tf = null;
	}
	else if (borde.equals("Esqueleto"))
	{
		p = pm.esqueletizar(e);
		titulo = "Esqueleto de " + titulo;
		tf = null;
	}
	else if (borde.equals("Cerco convexo"))
	{
		p = pm.cercar(e);
		titulo = "Cerco de " + titulo;
		tf = null;
	}
	else if (borde.equals("Poda"))
	{
		p = pm.podar(e);
		// niteraciones=pm.niteraciones;
		titulo = "Poda de " + titulo;
		tf = null;
	}
}
```

```
/**
 * Aplica una operación morfológica a una imagen en escala de grises
 *
 * @param e EE con el que vamos a realizar la operación
 * @param operacion String que especifica la operación a realizar
 */
public void operaMorfGrises(EE eg, String operacion)
{
	inicializaProcMorf();
	int[][] dst1 = new int[h][w];
	int[][] dst2 = new int[h][w];

	if (operacion.equals("Dilatación"))
	{
		// System.out.println("a dilatar");
		p = pm.dilatacionGrises(eg);
		titulo = "Dilatación de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Erosión"))
	{
		// System.out.println("a erosionar");
		p = pm.erosionGrises(eg);
		titulo = "Erosión de " + titulo;
		tf = null;
	}
	// Simplifican las imágenes, elimina picos positivos más estrechos que
	// el EE
	else if (operacion.equals("Apertura"))
	{
		// System.out.println("a abrir");
		p = pm.aperturaGrises(eg);
		titulo = "Apertura de " + titulo;
		tf = null;
	}
	// Simplifican las imágenes, elimina picos negativos más estrechos que
	// el EE
	else if (operacion.equals("Cierre"))
	{
		// System.out.println("a cerrar");
		p = pm.cierreGrises(eg);
		titulo = "Cierre de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Suavizado"))
	{
		// System.out.println("a suavizar");
		p = pm.aperturaGrises(eg);
		p = pm.cierreGrises(eg);
		titulo = "Suavizado de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Gradiente"))
	{
		// System.out.println("gradiente");
		dst1 = p;
		p = pm.dilatacionGrises(eg);
		dst2 = p;
		p = dst1;
		p = pm.erosionGrises(eg);
		dst1 = p;
		p = pm.resta(dst2, dst1);
		titulo = "Gradiente de " + titulo;
		tf = null;
	}
	else if (operacion.equals("Sombrero de copa"))
	{
		// System.out.println("sombrero");
		dst1 = p;
		p = pm.aperturaGrises(eg);
		p = pm.resta(dst1, p);
		titulo = "Sombrero de copa de " + titulo;
		tf = null;
	}
}
```

```
// --------MÉTODOS DEL CURSO RESTAURACIÓN DE IMÁGENES----------

    /**
     * Inicializamos el objeto restImg pasándole la imagen que existe en el momento.
     */
    public void inicializaRestImg()
    {
            restImg = new RestauracionImg(this);
    }

    /**
     * Devuelve el atributo restImg.
     *
     * @return el atributo restImg.
     */
    public RestauracionImg getRestImg()
    {
            return restImg;
    }

    /**
     * Aplica una distorsion geométrica sobre la imagen actual.
     * La imagen distorsionada pasa a ser la imagen actual
     * @param tipoDistorsion que se va a aplicar a la imagen actual
     */
    public void distorsionGeometrica(String tipoDistorsion)
    {
            inicializaRestImg();
            p = restImg.distorsionGeometrica(tipoDistorsion);
            titulo = "Distorsión " + tipoDistorsion + " de " + titulo;
            tf = null;
    }

    /**
     * Multiplica las dos imágenes que se pasan por parámetros
     * siendo la imagen resultante como producto de ambas la imagen
     * actual
     * @param img1 imagen1 que se quiere multiplicar
     * @param img2 imagen2 que se quiere multiplicar
     */
    public void multiplicaDosImagenes (Imagen img1, Imagen img2)
    {
            inicializaRestImg();
            p=restImg.multiplicaDosImagenes(img1,img2);
            titulo="Multiplicación de '" + img1.getTitulo() + "' y '" + img2.getTitulo() + "'";
            tf=null;
    }
}
```

8.2.2. Clase RestauracionImg

```
/**
 * Clase en la que se encuentran todos los métodos que implementan la
 * funcionalidad de los applets de Restauracion de Imágenes
 *
 * @author Ana Belén Alonso Martín
 * @version IMAGine 2.0 - 2008
 */
public class RestauracionImg implements DistorsionesGeometricasCtes
{
    /**
     * Valor de los píxeles de la imagen
     */
    private int[][] p;

    /**
     * Anchura de la imagen
     */
    private int w;

    /**
     * Altura de la imagen
     */
    private int h;
```

```
/**
 * Constructor de la clase que nos sirve para obtener el valor de los
 * atributos de nuestra clase necesarios para realizar las operaciones.
 *
 * @param imagen a la cual queremos aplicar cualquiera de los métodos
 * de esta clase.
 */
public RestauracionImg()
{
	p = null;
	h = 0;
	w = 0;
}

/**
 * Constructor de la clase que nos sirve para obtener el valor de los
 * atributos de nuestra clase necesarios para realizar las operaciones.
 *
 * @param imagen a la cual queremos aplicar cualquiera de los métodos
 * de esta clase.
 */
public RestauracionImg(Imagen imagen)
{
	p = imagen.getPixels();
	h = imagen.getHeight();
	w = imagen.getWidth();
}

/**
 * Multiplica el array de datos que representa el módulo de una imagen
 * a color contra una mascara
 * @param moduloTF array de datos que representa la TF de una imagen
 * @param masacara imagen con los datos de la máscara que vamos a multipliar contra la TF
 * @return array de datos resultado de multiplicar el moduloTF con la mascara
 */
public Array2D multiplicaDosModulos(Array2D moduloTF, ImagenColor mascara)
{
   Array2D inputAux = new Array2D(moduloTF);

   int[][] pixelesImgMascara = mascara.getPixelsBidimensional();
      int width = mascara.getWidth();
      int height = mascara.getHeight();

   for(int he=0;he<height;he++)
   {
     // positive side of axis
     for(int wi=0;wi<width;wi++)
     {
       if (ImagenColor.compRojoDeUnPixel(pixelesImgMascara[he][wi])<=0 &&
                                ImagenColor.compVerdeDeUnPixel(pixelesImgMascara[he][wi])<=0 &&
                                ImagenColor.compAzulDeUnPixel(pixelesImgMascara[he][wi])<=0)
       {
                        //Hacemos este cambio de coordenadas para que corresponan las de la mascara
                        //con las del modulo
                        if (wi>=width/2 && he>=height/2)
                        {
                                inputAux.values[he-height/2][wi-width/2]=new NumComplejo(0,0);
                        }
                        else if (wi<width/2 && he<height/2)
                        {
                                inputAux.values[height/2+he][width/2+wi]=new NumComplejo(0,0);
                        }
                        else if (wi>=width/2 && he<height/2)
                        {
                                inputAux.values[he+height/2][wi-width/2]=new NumComplejo(0,0);
                        }
                        else if (wi<width/2 && he>=height/2)
                        {
                                inputAux.values[he-height/2][wi+width/2]=new NumComplejo(0,0);
                        }
       }
     }
   }
   return inputAux;
}
```

```
/**
 * Realiza la multiplicación pixel a pixel de las dos imágenes que se pasan
 * por parámetros al métodos
 * @param img1 Imagen 1 que queremos multiplicar
 * @param img2 Imagen 2 que queremos multiplicar
 * @return pixeles de la imagen resultado de multiplicar las 2 que hemos pasado por parámetros al método
 */
public int[][] multiplicaDosImagenes(Imagen img1, Imagen img2)
{
	//Construimos un array de int que formarán los pixeles de la nueva
	// imagen
	int[][] pixelesImg = new int[img1.getHeight()][img1.getWidth()];
	//Obtenemos los pixeles de la primera img
	int[][] pixelesImg1 = img1.getPixels();
	//Obtenemos los pixeles de la segunda img
	int[][] pixelesImg2 = img2.getPixels();
	//Multiplicamos píxel a píxel la img1 con la img2 y asignamos el valor
	//resultante al array de int que representa los pixeles de la img
	// producto
	for (int i = 0; i < img1.getHeight(); i++)
	{
		for (int j = 0; j < img1.getWidth(); j++)
		{
			pixelesImg[i][j] = pixelesImg1[i][j] * pixelesImg2[i][j];
		}
	}
	return pixelesImg;
}

/**
 * Multiplica las dos imágenes que se pasan por parámetros pixel a pixel
 * @param img1 imagen1 que se quiere multiplicar
 * @param img2 imagen2 que se quiere multiplicar
 */
public ImagenColor multiplicaDosImagenes (ImagenColor img1, ImagenColor img2)
{
	//Construimos un array de int que formarán los pixeles de la nueva
	// imagen
	int[][] pixelesImg = new int[img1.getHeight()][img1.getWidth()];
	//Obtenemos los pixeles de la primera img
	int[][] pixelesImg1 = img1.getPixelsBidimensional();
	//Obtenemos los pixeles de la segunda img
	int[][] pixelesImg2 = img2.getPixelsBidimensional();
	//Multiplicamos píxel a píxel la img1 con la img2 y asignamos el valor
	//resultante al array de int que representa los pixeles de la img
	// producto
	for (int i = 0; i < img1.getHeight(); i++)
	{
		for (int j = 0; j < img1.getWidth(); j++)
		{
			pixelesImg[i][j] = pixelesImg1[i][j] * pixelesImg2[i][j];
			if (ImagenColor.compRojoDeUnPixel(pixelesImg[i][j])<=0 &&
				ImagenColor.compVerdeDeUnPixel(pixelesImg[i][j])<=0 &&
				ImagenColor.compAzulDeUnPixel(pixelesImg[i][j])<=0 &&
				ImagenColor.compAlphaDeUnPixel(pixelesImg[i][j])<=0)
			{
				pixelesImg[i][j]=ImagenColor.cambiaCompAlphaDeUnPixel(pixelesImg[i][j],255);
			}
		}
	}
	Image imageProducto=FuncionesImage.creaImage(img2.getWidth(),img1.getHeight(),pixelesImg);
	ImagenColor imagenProducto = new ImagenColor (imageProducto);
	imagenProducto.setTitulo ("Multiplicación de '" + img1.getTitulo() + "' y '" + img2.getTitulo() + "'");
	return imagenProducto;
}

/**
 * Método que aplica una distorsion geométrica a la imagen que hemos pasado al contructor de esta clase
 * @param tipoDistorsion que queremos aplicar a la imagen
 * @return pixeles de la imagen distorsionada geométricamente
 */
public int[][] distorsionGeometrica(String tipoDistorsion)
{

	int[][] pixeles = new int[h][w];

	double pendiente;
	double reescalado;
	double desplazamientoVertical;
	double desplazamientoHorizontal;
	int margenSuperior;
```

```
int margenInferior;
int margenIzquierdo;
int margenDerecho;
int coorX;
int coorY;

//DISTORSIÓN: BARREL
if (tipoDistorsion.equals(TIPOS_DISTORSION[0]))
{
	reescalado = 1.0;
	pendiente = 8;
	margenSuperior = 10;
	margenInferior = 15;
	margenIzquierdo = 20;
	margenDerecho = 0;

	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{
			//1) Obtenemos el valor de la transformación de las
			// coordenadas:

			//Esquina superior izda
			if (i <= pixeles.length * 3 / 6 && j <= pixeles.length * 3 / 6)
			{
				desplazamientoVertical = 0;
				desplazamientoHorizontal = 13;
				coorX = (int) ((i + j / pendiente) * reescalado + desplazamientoVertical)
						- margenSuperior;
				coorY = (int) ((j + i / pendiente) * reescalado + desplazamientoHorizontal)
						- margenIzquierdo;
			}
			//Esquina inferior izda
			else if (i > pixeles.length * 3 / 6 && j < pixeles.length * 3 / 6)
			{
				desplazamientoVertical = 0;
				desplazamientoHorizontal = 45;
				coorX = (int) ((i - j / pendiente) * reescalado + desplazamientoVertical)
						+ margenInferior;
				coorY = (int) ((j - i / pendiente) * reescalado + desplazamientoHorizontal)
						- margenIzquierdo;
			}
			//Esquina superior dcha
			else if (i < pixeles.length * 3 / 6  && j > pixeles.length * 3 / 6)
			{
				desplazamientoVertical = 33;
				desplazamientoHorizontal = 13;
				coorX = (int) ((i - j / pendiente) * reescalado + desplazamientoVertical)
						- margenSuperior;
				coorY = (int) ((j - i / pendiente) * reescalado + desplazamientoHorizontal)
						+ margenDerecho;
			}
			//Esquina inferior dcha
			else
			{
				desplazamientoVertical = -34;
				desplazamientoHorizontal = -18;
				coorX = (int) ((i + j / pendiente) * reescalado + desplazamientoVertical)
						+ margenInferior;
				coorY = (int) ((j + i / pendiente) * reescalado + desplazamientoHorizontal)
						+ margenDerecho;
			}

			//2) Asignamos el valor al pixel de las nuevas coordenadas:

			//Solo cogemos las coordenadas que no se salen del
			// contenedor de la imagen
			if (coorY < pixeles.length && coorY > 0
				&& coorX < pixeles.length && coorX > 0)
			{
				pixeles[i][j] = p[coorX][coorY];
			}
			else
			{
				pixeles[i][j] = 0;
			}
		}
	}

}
```

```
//DISTORSIÓN: PIN CUSHION
else if (tipoDistorsion.equals(TIPOS_DISTORSION[1]))
{
	reescalado = 0.95;
	pendiente = 8;
	margenSuperior = 20;
	margenInferior = 45;
	margenIzquierdo = -10;
	margenDerecho = 5;

	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{

			//1) Obtenemos el valor de la transformación de las
			// coordenadas:

			//Esquina superior izda
			if (i <= pixeles.length * 3 / 6
				&& j <= pixeles.length * 3 / 6)
			{
				desplazamientoVertical = 31;
				desplazamientoHorizontal = 0;
				coorX = (int) ((i - j / pendiente) * reescalado + desplazamientoVertical)
						- margenSuperior;
				coorY = (int) ((j - i / pendiente) * reescalado + desplazamientoHorizontal)
						- margenIzquierdo;
			}
			//Esquina inferior izda
			else if (i > pixeles.length * 3 / 6
					&& j < pixeles.length * 3 / 6)
			{
				desplazamientoVertical = -31;
				desplazamientoHorizontal = -31;
				coorX = (int) ((i + j / pendiente) * reescalado + desplazamientoVertical)
						+ margenInferior;
				coorY = (int) ((j + i / pendiente) * reescalado + desplazamientoHorizontal)
						- margenIzquierdo;
			}
			//Esquina superior dcha
			else if (i < pixeles.length * 3 / 6
					&& j > pixeles.length * 3 / 6)
			{
				desplazamientoVertical = 0;
				desplazamientoHorizontal = 0;
				coorX = (int) ((i + j / pendiente) * reescalado + desplazamientoVertical)
						- margenSuperior;
				coorY = (int) ((j + i / pendiente) * reescalado + desplazamientoHorizontal)
						+ margenDerecho;
			}
			//Esquina inferior dcha
			else
			{
				desplazamientoVertical = 0;
				desplazamientoHorizontal = 31;
				coorX = (int) ((i - j / pendiente) * reescalado + desplazamientoVertical)
						+ margenInferior;
				coorY = (int) ((j - i / pendiente) * reescalado + desplazamientoHorizontal)
						+ margenDerecho;
			}

			//2) Asignamos el valor al pixel de las nuevas coordenadas:

			//Solo cogemos las coordenadas que no se salen del
			// contenedor de la imagen
			if (coorY < pixeles.length && coorY > 0
					&& coorX < pixeles.length && coorX > 0)
			{
				pixeles[i][j] = p[coorX][coorY];
			}
			else
			{
				pixeles[i][j] = 0;
			}
		}
	}

}
```

```
//DISTORSIÓN: VERTICAL
else if (tipoDistorsion.startsWith("distorsion_vertical_"))
{

	if (tipoDistorsion.equals(TIPOS_DISTORSION[2]))  //"distorsion_vertical_1"
	{
		pendiente = 7;
		reescalado = 0.85;
		desplazamientoVertical = 0;
	}
	else if (tipoDistorsion.equals(TIPOS_DISTORSION[3])) //"distorsion_vertical_2"
	{
		pendiente = 1.5;
		reescalado = 0.6;
		desplazamientoVertical = 0;
	}
	else if (tipoDistorsion.equals(TIPOS_DISTORSION[4])) //"distorsion_vertical_3"
	{
		pendiente = -10;
		reescalado = 0.9;
		desplazamientoVertical = 25;
	}
	else
	{
		//Ponemos valores por defecto
		pendiente = 1;
		reescalado = 1;
		desplazamientoVertical = 0;
	}

	for (int i = 0; i < h; i++)
	{
		for (int j = 0; j < w; j++)
		{
			//Aqui aplicamos la transformacion lineal de las
			// coordenadas
			coorX = (int) ((i + j / pendiente) * reescalado + desplazamientoVertical);
			coorY = j;
			//Solo cogemos las coordenadas que no se salen del
			// contenedor de la imagen
			if (coorY < pixeles.length && coorY > 0
			    && coorX < pixeles.length && coorX > 0)
			{
				pixeles[coorX][coorY] = p[i][j];
			}
		}
	}
}

//DISTORSIÓN: HORIZONTAL
else if (tipoDistorsion.startsWith("distorsion_horizontal_"))
{

	if (tipoDistorsion.equals(TIPOS_DISTORSION[5])) //"distorsion_horizontal_1"
	{
		pendiente = 7;
		reescalado = 0.85;
		desplazamientoHorizontal = 0;
	}
	else if (tipoDistorsion.equals(TIPOS_DISTORSION[6])) //"distorsion_horizontal_2"
	{
		pendiente = 4;
		reescalado = 0.8;
		desplazamientoHorizontal = 0;
	}
	else if (tipoDistorsion.equals(TIPOS_DISTORSION[7])) //"distorsion_horizontal_3"
	{
		pendiente = -10;
		reescalado = 0.9;
		desplazamientoHorizontal = 25;
	}
	else
	{
		//Ponemos valores por defecto
		pendiente = 1;
		reescalado = 1;
		desplazamientoHorizontal = 0;
	}

	for (int i = 0; i < h; i++)
	{
```

```
                                for (int j = 0; j < w; j++)
                                {
                                        //Aqui aplicamos la transformacion lineal de las
                                        // coordenadas
                                        coorX = i;
                                        coorY = (int) ((j + i / pendiente) * reescalado + desplazamientoHorizontal);
                                        //Solo cogemos las coordenadas que no se salen del
                                        // contenedor de la imagen
                                        if (coorY < pixeles.length && coorY > 0
                                                && coorX < pixeles.length && coorX > 0)
                                        {
                                                pixeles[coorX][coorY] = p[i][j];
                                        }
                                }
                        }
                }
                return pixeles;
        }
}
```

9. ÍNDICE DE FIGURAS

9. ÍNDICE DE FIGURAS

Este índice de imágenes está dividido en los mismos capítulos de los que consta esta memoria, de tal forma que en cada sección se incluyen las figuras que corresponden a dicho capítulo en la memoria.

Por cada figura se incluye el número identificativo, una breve descripción y la página de la memoria donde se puede encontrar.

Nº Figura Descripción Página

1. Introducción

Figura 1.1: *Historia de IMAGine: versiones, autores y contribuciones* 15

2. Estado del arte

Figura 2.1: *Búsqueda en Google de "Curso interactivo Tratamiento Digital de Imagen" con posición de IMAGine* 24

Figura 2.2: *Curso Trat. Digital de Imágenes de la Universidad de Málaga* 25

Figura 2.3: *Curso Trat. Digital de Imágenes de la Universidad de Málaga → Applets* 28

Figura 2.4: *Curso 'HIPR2: Image Processing Learning Resources' de la Universidad de Edimburgo* 28

Figura 2.5: *Curso HIPR2 → Índice Temas (Worksheets)* 29

Figura 2.6: *Curso HIPR2 → Navegación por el Índice de Temas* 30

Figura 2.7: *Curso HIPR2 → Applets* 33

Figura 2.8: *Curso Trat. Digital de Imágenes de la Universidad de la Coruña* 34

Figura 2.9: *Curso Trat. Digital de Imágenes de la Universidad de la Coruña → Applets* 35

Figura 2.10: *Aplicación procesado de Imágenes de UPIICSA* 38

Figura 2.11: *Análisis cursos similares a IMAGine: cuadro resumen puntuaciones otorgadas* 39

3. Teoría sobre tratamiento digital de imágenes

Figura 3.1: *Modelos de color* 44

Figura 3.2: *Diagrama Digitalización de Imágenes* 45

Figura 3.3: *Modelos de color* 45

Figura 3.4: *Histograma de una imagen* 47

Figura 3.5: *Brillo de una imagen* 47

Figura 3.6: *Contraste de una imagen* 48

Figura 3.7: *Ejemplo mejora del contraste* 50

Figura 3.8: *Modelo cancelación de interferencias sinusoidales* 51

Figura 3.9: *Eliminación de ruido impulsivo con filtro paso bajo y de mediana* 51

Figura 3.10: *Eliminación de ruido gaussiano con filtro paso bajo y de mediana* 52

Figura 3.11: *Modelo para la transformación geométrica de una imagen* 52

Figura 3.12: *Ejemplos distorsiones geométricas* 53

Figura 3.13: *Modelo de decoloración típico de una imagen a color deteriorada por el paso del tiempo* 53
Figura 3.14: *Ejemplos restauración imágenes a color* 54
Figura 3.15: *Modelo de degradación de imágenes* 55
Figura 3.16: *Modelo de degradación de imágenes. ¿Cómo obtener la función de degradación (PSF)?* 56
Figura 3.17: *Casos típicos de distorsión (desenfoque, movimiento e imagen doble)* 56
Figura 3.18: *Modelo estocástico del proceso de degradación de imágenes (Filtro de Wiener)* 57
Figura 3.19: *Modelo de filtro lineal en el que se basa el Filtro de Wiener* 57
Figura 3.20: *Ejemplos técnicas restauración (filtrado inverso y filtro de Wiener) para imágenes con distorsiones de tipo desenfoque, movimiento e imagen doble* 58

4.1. Descripción de la herramienta: Aplicación Web

Figura 4.1: *Apariencias de las versiones anteriores y la actual de IMAGine* 63
Figura 4.2: *Menús de los cursos de IMAGine* 64
Figura 4.3: *Mapa esquemático de la Web de IMAGine* 66
Figura 4.4: *'Portada' de IMAGine* 67
Figura 4.5: *Apartado 'Funcionamiento del curso' de IMAGine* 67
Figura 4.6: *Apartado 'Índice Applets' y ventana emergente con todos los applets de IMAGine* 68
Figura 4.7: *Apartado 'Enlaces' de IMAGine* 68
Figura 4.8: *Apartado 'Autores' de IMAGine* 69
Figura 4.9: *Apariencia de un 'Curso' de IMAGine* 69
Figura 4.10: *Apariencia de una 'Autoevaluación' de IMAGine* 71
Figura 4.11: *Etiquetas validaciones estándares de la W3C* 72
Figura 4.12: *Esquema secciones de las páginas XHTML con la teoría de los cursos* 84
Figura 4.13: *Estructura de directorios de un curso de IMAGine* 92

4.2. Descripción de la herramienta: Applets

Figura 4.14: *Clases de IMAGine antes de este proyecto* 95
Figura 4.15: *Estructura clases de IMAGine antes de este proyecto* 96
Figura 4.16: *Estructura clases de IMAGine en este proyecto* 98
Figura 4.17: *Menú desplegable al pinchar con el botón derecho sobre una imagen en escala de grises º* 100
Figura 4.18: *Applet Operaciones Puntuales* 101
Figura 4.19: *Ejemplo de operación puntual (modificación del contraste)* 101
Figura 4.20: *Applet Filtrado basado en máscaras* 102
Figura 4.21: *Applet Importancia componentes de baja frecuencia* 102
Figura 4.22: *Ejemplo de importancia de las componentes de baja frecuencia* 103
Figura 4.23: *Applet Filtro de mediana* 104
Figura 4.24: *Ejemplo de la utilización del filtro de mediana* 104
Figura 4.25: *Applet Rotación de imágenes y su efecto en la DFT* 105

Figura 4.26: *Ejemplo de rotación de imágenes y su efecto en la DFT* 105
Figura 4.27: *Applet Contribución de las componentes frecuenciales en la transformada de Fourier* 106
Figura 4.28: *Applet Importancia de la fase (síntesis con módulo unidad o fase cero)* 107
Figura 4.29: *Applet Experimento del Oppenheim* 108
Figura 4.30: *Applet Promedio de módulos y fases* 109
Figura 4.31: *Applet Operaciones morfológicas sobre imágenes binarias* 110
Figura 4.32: *Algoritmo Hit or Miss* 110
Figura 4.33: *Applet Transformación Hit or Miss* 111
Figura 4.34: *Applet Algoritmos morfológicos* 112
Figura 4.35: *Applet Operaciones morfológicas sobre imágenes en escala de grises* 113
Figura 4.36: *Applet Ecualización del histograma* 114
Figura 4.37: *Applet Cancelación de Interferencias Sinusoidales* 115
Figura 4.38: *Applet Eliminación de ruido* 116
Figura 4.39: *Ejemplo técnicas eliminación de ruido* 116
Figura 4.40: *Applet Distorsiones Geométricas* 117
Figura 4.41: *Applet Imágenes a color* 118
Figura 4.42: *Ejemplos operaciones sobre imágenes a color* 118
Figura 4.43: *Package BaseImagenes. Clases que se ocupan de la base de imágenes* 119
Figura 4.44: *Programas para gestionar la bases de imágenes de IMAGine (Func. Clase Editor)* 121
Figura 4.45: *Panel 'Eliminar imágenes' de la BBII. (Func. Clase BaseImg_DeletePanel)* 122
Figura 4.46: *Panel 'Cargar imagen' en el applet. (Func. Clase BaseImg_CargaPanel)* 123
Figura 4.47: *Package comun. Clases que contiene la funcionalidad básica de IMAGine* 124
Figura 4.48: *Paneles emergentes de la clase ImagenCanvas_PanelesSubmenu* 127
Figura 4.49: *Paneles emergentes de la clase ImagenColorCanvas_PanelesSubmenu* 131
Figura 4.50: *Packages de los applets de los cursos. Clases con la implementación de los applets* 135
Figura 4.51: *Estructura interna de clases que heredan de IMAGineApplet (tienen sufijo 'Applet')* 136

5. Estadísticas de Tráfico Recibido

Figura 5.1: *Histórico de visitas a IMAGine desde Enero 2007 hasta Febrero 2008* 145
Figura 5.2: *Visitas a IMAGine en Febrero 2008 (fecha examen de la asignatura)* 146
Figura 5.3: *Países de procedencia de las visitas a IMAGine* 146
Figura 5.4: *Estadísticas de navegadores utilizados en la navegación por IMAGine* 147
Figura 5.5: *Estadísticas de resoluciones de pantalla utilizadas en la navegación por IMAGine* 147
Figura 5.6: *Estadísticas de velocidad de conexión al conectarse a IMAGine* 148

7. Líneas Futuras

Figura 7.1: *Temario asignatura Tratamiento Digital de Imagen en la Universidad Carlos III* 155
Figura 7.2: *Ejemplo práctico de un posible ejercicio para el applet de autoevaluación* 158

10. BIBLIOGRAFÍA

10. BIBLIOGRAFÍA

Esta bibliografía está dividida en los mismos capítulos de los que consta esta memoria, de tal manera que en cada uno de ellos se incluya la bibliografía consultada para su elaboración.

También se incluye bibliografía adicional para ampliar información sobre algunos de los temas tratados en estos capítulos.

1. INTRODUCCIÓN

[Bib. 1.1] Web de IMAGine
http://www.tsc.uc3m.es/imagine

2. ESTADO DEL ARTE

• E-learning:

[Bib. 2.1] Definición en la wikipedia de e-learning:
http://es.wikipedia.org/wiki/E-learning

[Bib. 2.2] Artículo de Julio Cabrero "Bases pedagógicas del e-learning" de la Revista de Universidad y Sociedad del Conocimiento.
http://redalyc.uaemex.mx/redalyc/pdf/780/78030102.pdf

[Bib. 2.3] Plataformas para teleeducación: disponibles enlaces en
http://es.wikipedia.org/wiki/E-learning#Plataformas_de_e-learning
-*Plataformas libres (Código abierto):* ATutor, Bodington, Claroline, Dokeos, KEWL, ILIAS, .LRN, LON-CAPA, Moodle, Sakai Project, LogiCampus, etc.
-*Plataformas No Libres (Privadas):* NETcampus, Skillfactory, ANGEL Learning, VerticeLearning, Blackboard, Brihaspati, Desire2Learn, Edumate, FirstClass, Knowledge Forum, Authorware, Plataforma Mediáfora, Scholar360, WebCT, Litmos, CyberExtension, Aula de futuro, etc.

• URL de Webs con Cursos Interactivos de Tratamiento Digital de Imágenes:

[Bib. 2.4] Curso interactivo de Trat. Digital de Imagen (Universidad de Málaga):
http://campusvirtual.uma.es/tdi/

[Bib. 2.5] Curso 'HIPR2: Image Processing Lerning Resources' (Edimburgo University):
http://homepages.inf.ed.ac.uk/rbf/HIPR2/

[Bib. 2.6] Curso 'Técnicas de Procesado de Imagen' (Universidad de la Coruña):
http://www.des.udc.es/~adriana/TercerCiclo/CursoImagen/curso/web/Indice.html

• URL de Webs con Cursos Teóricos de Tratamiento Digital de Imágenes:

[Bib. 2.7] Curso FIP: curso teórico sobre procesamiento de imágenes (Universidad Delf University of Technology) [inglés]
http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html

[Bib. 2.8] 'DIP: Digital Image Processing with Khoros': Curso Teórico, con ejercicios y ejemplos, y prácticas de laboratorio con Khoros sobre Tratamiento Digital de Imágenes (Universidad de Campinas) [inglés]
http://www.dca.fee.unicamp.br/dipcourse/index.html

[Bib 2.9] Teoría (en inglés) sobre Tratamiento digital de Imágenes (Universidad de Iowa)
http://www.icaen.uiowa.edu/~dip/LECTURE/contents.html

[Bib. 2.10] Curso teórico de procesado de imágenes digitales y Topología digital [interés Tema 4 que trata sobre Procesado Morfológico] (Universidad de Sevilla)
http://www.sav.us.es/formaciononline/asignaturas/asigpid/

[Bib. 2.11] Curso teórico de procesado de imágenes digitales [nociones básicas sobre imagen digital, formatos de imagen, proceso y análisis de imágenes...] (Universidad de Oviedo)
http://wellpath.uniovi.es//es/contenidos/seminario/tutorialpdi/html/index.htm

• URL de Applets/Aplicaciones sobre Tratamiento Digital de Imágenes:

[Bib. 2.12] Descarga aplicación para ejecutar en local sobre Procesado de Imágenes (UPIICSA):
http://upiicsa.robotica.googlepages.com/ImageProcessorSample.zip

3. TEORÍA SOBRE TRATAMIENTO DIGITAL DE IMÁGENES

• Teoría Tratamiento Digital de Imágenes:

[Bib. 3.1] Libro de referencia sobre procesado de imágenes:
· Rafael C. Gonzalez, Richard E. Woods. *Digital Image Processing* / *Tratamiento Digital de Imágenes*. Ed. Addison-Wesley.

* Disponible también en Google Books:
chttp://books.google.es/books?id=TQT6DF3_YgkC&pg=PA6&dq=tratamiento+digital+de+imagen&sig=eYOWtGLXUf3x50SApMC5vYFvay0#PPP1,M1

[Bib. 3.2] Libro de fundamentos sobre procesado de imágenes:
· Anil K. Jain. *Fundamentals of Digital Image Processing*. Ed. Prentice Hall.

[Bib. 3.3] Libro con teoría sobre procesado digital de imágenes:
· W.K. Prat. *Digital Image Processing*. John Wiley and sons Ed.

[Bib. 3.4] Manual sobre procesado de imágenes:
· B. Jähne. *Practical Handbook on Image Processing*. Ed. CRC.

[Bib. 3.5] Libro básico sobre procesado de imágenes:
· J.C. Russ. *The Image Procesiing Handbook.* Ed. CRC Press.

[Bib. 3.6] Páginas webs que tratan el Tratamiento Digital de Imágenes:
http://es.wikipedia.org/wiki/Procesamiento_digital_de_im%C3%A1genes
http://www.scribd.com/doc/331892/Procesamiento-Digital-de-Imagenes
http://wellpath.uniovi.es//es/contenidos/seminario/tutorialpdi/html/index.ht

[Bib. 3.7] Documentación en la web específica sobre Restauración de Imágenes:
http://es.wikipedia.org/wiki/Restauraci%C3%B3n_de_imagen
http://www.iua.upf.es/~mbertalmio//bertalmi.pdf
http://alojamientos.us.es/gtocoma/pid/pid10/Restauracion.htm

4.1. DESCRIPCIÓN DE LA HERRAMIENTA: APLICACIÓN WEB

• Accesibilidad Web

[Bib. 4.1] Resumen de la enciclopedia virtual Wikipedia sobre accesibilidad web:
http://es.wikipedia.org/wiki/Accesibilidad_web

[Bib. 4.2] Guía breve sobre accesibilidad web publicada por el W3C (organismo internacional que promueve estándares y recomendaciones en materia del web):
http://www.w3c.es/divulgacion/guiasbreves/Accesibilidad

[Bib. 4.3] Artículo de la revista 'No Solo Usabilidad journal' sobre accesibilidad web. Autores Yusef Hassan Montero y Fco J. Martín Fdez (Univ Granada).
http://www.nosolousabilidad.com/articulos/accesibilidad.htm

[Bib. 4.4] Libro sobre accesibilidad de los contenidos web, con las pautas principales para conseguir una web accesible:
· Pablo Lara Navarra. *La accesibilidad de los contenidos web.* Ed. UOC.

• HTML, XHTML y Hojas de estilo CSS:

[Bib. 4.5] Información oficial sobre las especificaciones de estos estándares de programación:
http://www.w3.org

[Bib. 4.6] Validador para ver si una página sigue las especificaciones del estándar:
http://validator.w3.org/

[Bib. 4.7] Resúmenes de la enciclopedia virtual Wikipedia sobre HTML, XHTML y CSS:
http://es.wikipedia.org/wiki/HTML
http://es.wikipedia.org/wiki/XHTML
http://es.wikipedia.org/wiki/Css

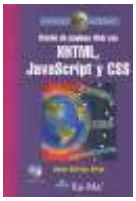
[Bib. 4.8] Guías breves sobre XHTML y Hojas de estilo CSS publicadas por el W3C:
http://www.w3c.es/divulgacion/guiasbreves/XHTML
http://www.w3c.es/Divulgacion/Guiasbreves/HojasEstilo

[Bib. 4.9] Tutoriales en Internet sobre HTML y XHTML:
http://www.programacion.net/html/tutorial/xhtml/
http://www.programacion.com/html/tutorial/curso/19/
http://www.cristalab.com/tutoriales/143/tutorial-basico-de-xhtml

[Bib. 4.10] Tutoriales en Internet sobre Hojas de estilo CSS:
Tutoriales generales:
http://www.w3schools.com/css/
http://www.webestilo.com/css/css00.phtml
Para que funcionen hojas de estilo alternativas:
http://www.kusor.net/traducciones/ala.es/alternate/
Menús desplegables:
http://www.jlvelazquez.net/accesibilidad/menusdesplegables.asp#
http://www.tunait.com/javascript/index.php?s=generadormenu
http://www.programacion.com/html/articulo/tw_menus1/
http://www.webmasterlibre.com/2006/05/27/menus-desplegables-con-javascript/

[Bib. 4.11] Símbolos (caracteres) y letras griegas (para las fórmulas) en XHTML:
http://ascii.cl/es/codigos-html.htm
http://www.w3.org/TR/html401/sgml/entities.html#h-24.3.1
http://es.wikipedia.org/wiki/Alfabeto_griego

[Bib. 4.12] Libro con información y multitud de ejemplos sobre estos tres estándares. Es una herramienta de gran utilidad para diseñar una buena página web, añadiendo a las mismas interactividad y contenidos multimedia.
· Juan Carlos Orós Cabello. *Diseño de páginas Web con XHTML, JavaScript y CSS*. Ed. Ra-Ma



[Bib. 4.13] Apuntes Tema 1 de la asignatura Presentación Multiplataforma.
· Celeste Campos. Apuntes Tema 1 Presentación Multiplataforma de 3º ITT Sonido e Imagen. UC3M Madrid 2005/2006.

• Javascript:

[Bib. 4.14] Resumen de la enciclopedia virtual Wikipedia sobre JavaScript:
http://es.wikipedia.org/wiki/JavaScript

[Bib. 4.15] Guía breve sobre JavaScript con algunos ejemplos:
http://www.webestilo.com/javascript/

[Bib. 4.16] Libro con información sobre javascript y css:
· Juan Carlos Orós Cabello. *Diseño de páginas Web interactivas con JavaScript y CSS: navegar en Internet*. Ed. Ra-Ma.

[Bib. 4.17] Libro con toda la información sobre JavaScript:
· Thomas A. Powell. *JavaScript: manual de referencia*. Ed. McGraw-Hill.

[Bib. 4.18] Libro con información general, trucos y ejemplos prácticos sobre JavaScript:
· Johannes Gamperl. *El gran libro de JavaScript: aspectos generales, trucos y consejos, soluciones prácticas, referencia del lenguaje*. Ed. Marcombo.

4.2. DESCRIPCIÓN DE LA HERRAMIENTA: APPLETS

• **Java:**

[Bib. 4.19] Libro de introducción a Java:
· Jesús Sánchez Allende. Java 2: iniciación y referencia. Ed. McGraw-Hill Interamericana de España.

[Bib. 4.20] Tutoriales de Java en Internet:
http://www.wikilearning.com/tutorial/tutorial_de_java/3938
http://www.ulpgc.es/otros/tutoriales/java/Intro/tabla.html

[Bib. 4.21] Tutoriales del API 'Java Advanced Imaging' de Java para tratamiento de imágenes:
http://java.sun.com/developer/onlineTraining/javaai/jai/index.html

[Bib. 4.22] Libro de tratamiento de imágenes con Java: contiene ejemplos sobre tratamiento digital de imágenes (procesado, efectos, …) con Java. [Capítulo 5: Imágenes con Java2D]:

· Sergio Gálvez Rojas, Manuel Alcalde García, Miguel Ángel Mora Mata. *Java a tope: Java2D. Cómo tratar con Java figuras, imágenes y texto en 2D*. Ed. UMA (Univ. De Málaga)

* Disponible también el Google Books:
http://books.google.es/books?id=M6reV4TGyIQC&printsec=frontcover

[Bib. 4.23] Breves tutoriales sobre como construir applets en Java, con los métodos principales y algún ejemplo:
http://www.ulpgc.es/otros/tutoriales/java/Cap2/escribir.html
http://web.cica.es/formacion/JavaTut/Cap2/holamapp.html

5. ESTADÍSTICAS DE TRÁFICO RECIBIDO

[Bib. 5.1] Google Analitycs:
http://www.google.com/analytics/es-ES